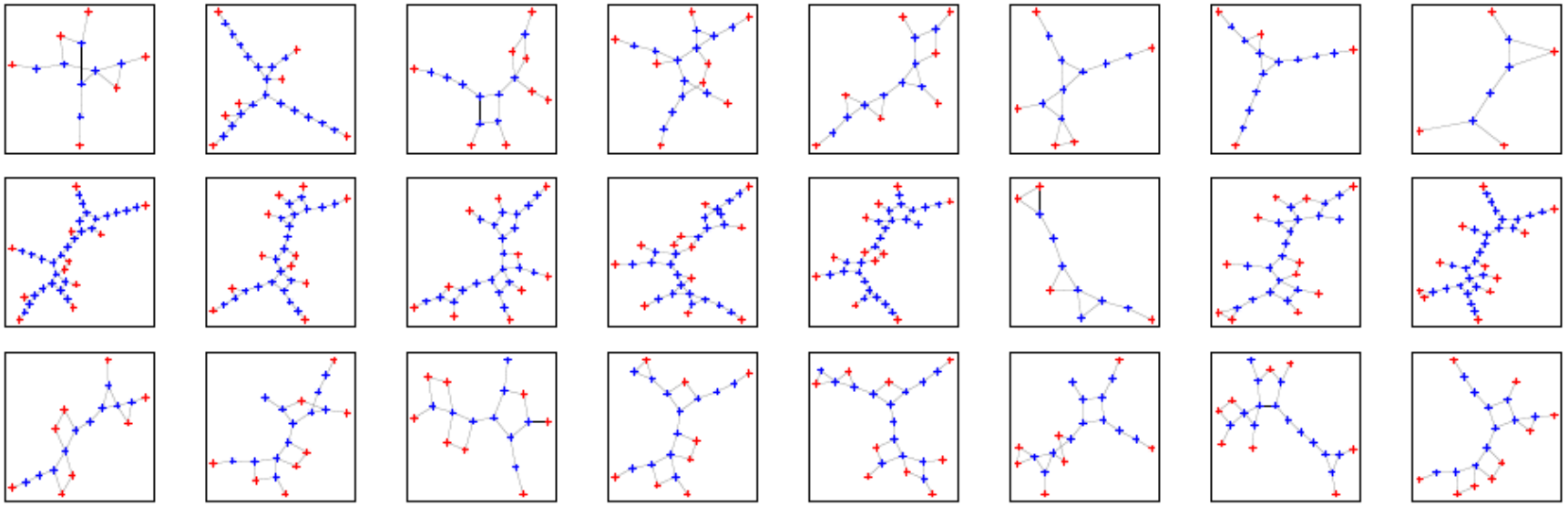
New Directions in Statistical Graph Mining

Max Planck Institute for Biological Cybernetics

Koji Tsuda

Joint work with Hiroto Saigo, Sebastian Nowozin, Nicole Kraemer, Takeaki Uno and Taku Kudo

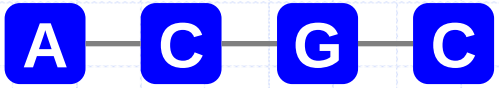
Graphs and itemsets



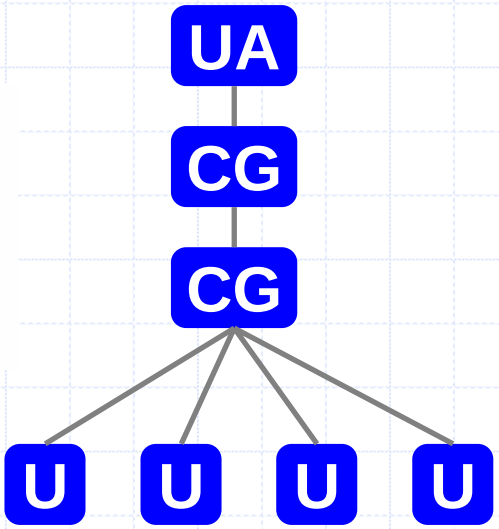
(40F,41L,43E,210W,211K,215Y)
(43Q,75M,122E,210W,211K)
(75I, 77L, 116Y,151M,184V)

Graph Structures in Biology

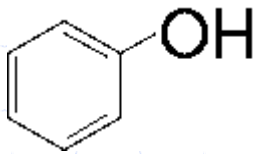
- DNA Sequence

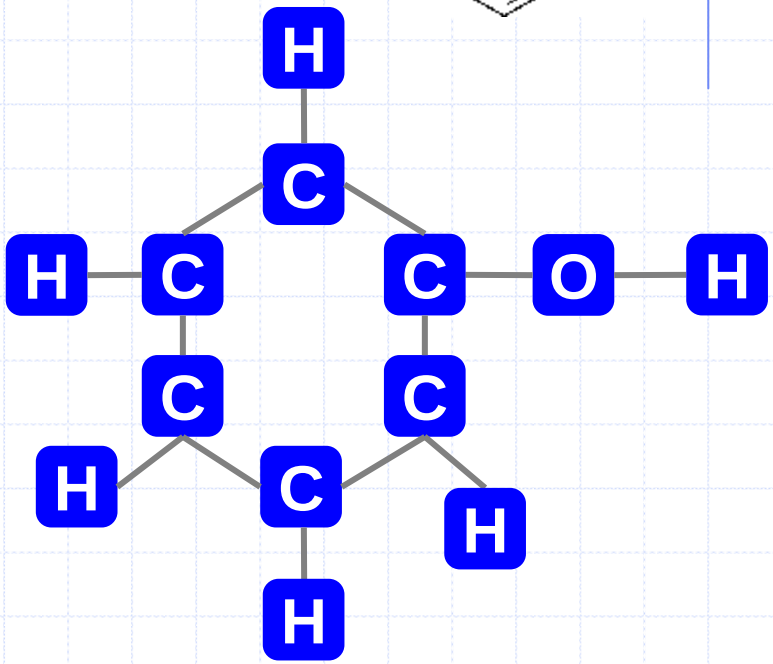


- RNA

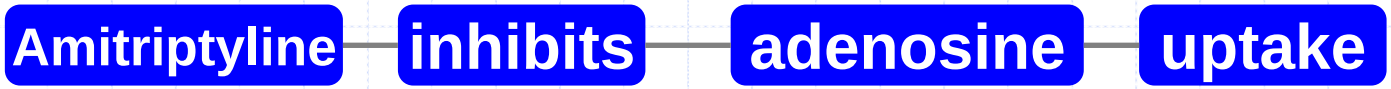


- Compounds





- Texts in literature



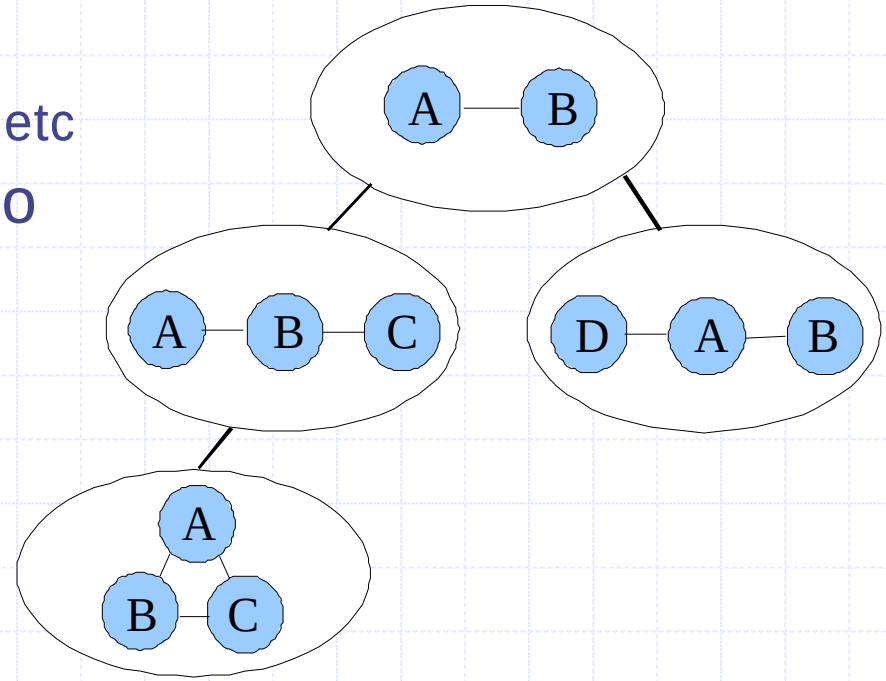
Structurization of statistical methods

- Statistical methods for numerical vectors
  - SVM, Boosting, Clustering, Regression, PCA, CCA..
- **Structurization**: Modify the algorithm so that it accepts structured data as input

- Recipe:
  - If the algorithm has a forward feature selection step, replace it with a top-k pattern mining algorithm
  - If not, the algorithm has to be engineered to have a forward feature selection step

Pattern Mining Algorithms

- Pattern: Substructure
  - Subset, subtree, subgraph etc
- Combinatorial algorithm to enumerate all patterns satisfying a criterion
  - Frequency in database
  - Weighted frequency
  - Top-k patterns for a given evaluation function
- See a textbook for data mining (e.g., Han and Kamber, 2002)



Naïve Structurization

- Apply frequent pattern mining and create the complete feature space
- Then, apply a statistical method

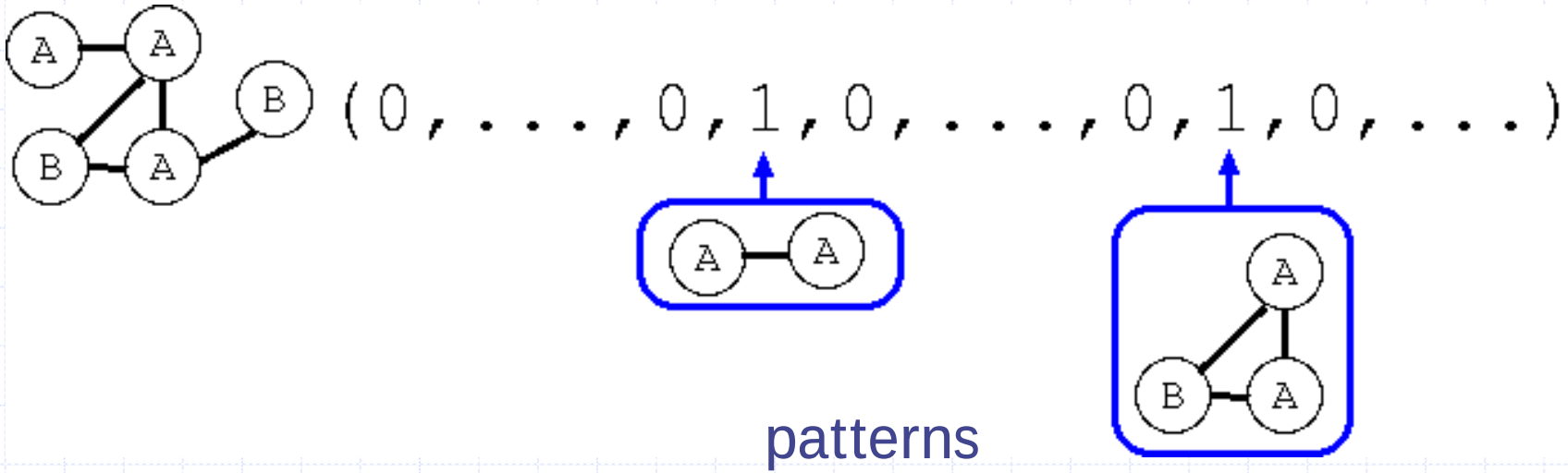
- Too many patterns!
- It is likely to miss important features if frequency threshold (minimum support) is set high
- Memory overflow
- See tutorial by Xifeng Yan (KDD08)

Progressive collection of patterns

- Most statistical methods are iterative
  - Boosting, PLS, Clustering etc
- In each iteration, call a pattern mining algorithm to collect a few patterns
  - Discover the optimal patterns based on the current parameter

Feature space is gradually constructed

- 0/1 vector of *pattern* indicators
- Dimensionality grows slightly in each iteration



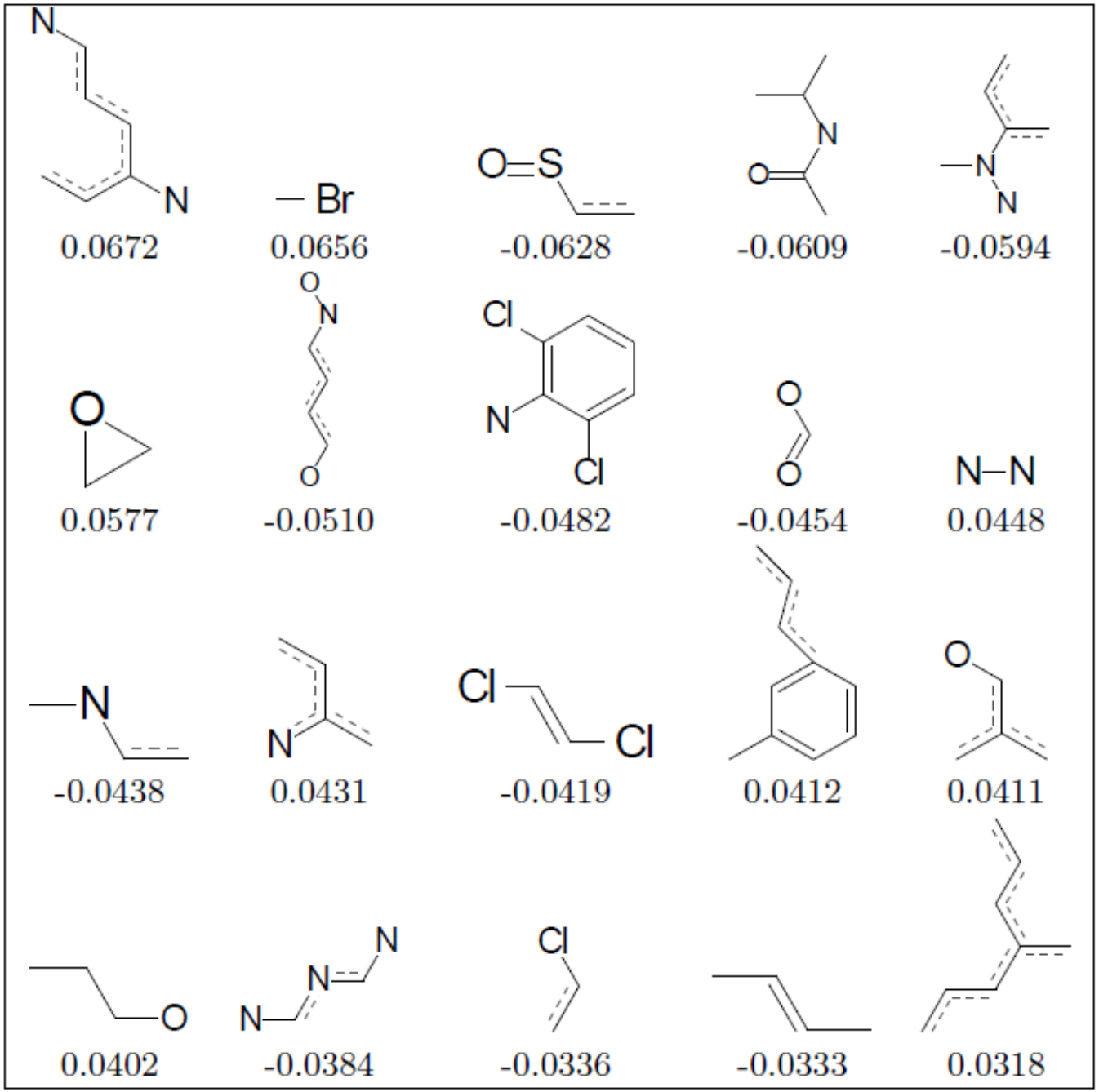
Structurization examples from my group

Method	Objects	Learning	Mining	Publication
gBoost	Graphs	L1SVM, LASSO	gspan	Machine Learning
gLARS	Graphs	LARS	gspan	ICML07
pBoost	Sequences	LPBoost	prefixspan	ICCV07
iBoost	Itemsets	LPBoost	LCM	Bioinformatics
graph EM	Graphs	Binomial Mixture	gspan	ICML06
graph DP	Graphs	DP Mixture	gspan	SDM08

gBoost (Saigo et al., 2006, 2008)

- Graph Classification: L1-SVM + Graph Mining
- Graph Regression: LASSO + Graph Mining

Method	Mutag		CAS		CPDB		AIDS (CAvsCM)	
	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC
Gaston [23]	-	-	0.79	-	-	-	-	-
MOLFEA [17]	-	-	-	-	0.785	-	-	-
CPM [4]	-	-	0.801	-	0.760	-	**0.832**	-
MGK	0.808	0.901	0.771	0.763	0.765	0.756	0.762	0.760
freqSVM	0.808	0.906	0.773	0.843	0.778	0.845	0.782	**0.808**
gBoost	**0.852**	**0.926**	**0.825**	**0.889**	**0.788**	**0.854**	0.802	0.774


0.0672
0.0656
-0.0628
-0.0609
-0.0594
0.0577
-0.0510
-0.0482
-0.0454
0.0448
-0.0438
0.0431
-0.0419
0.0412
0.0411
0.0402
-0.0384
-0.0336
-0.0333
0.0318

Related papers from other groups

- *Stream Prediction Using a Generative Model Based on Frequent Episodes in Event Sequences*. Srivatsan Laxman, Vikram Tankasali, Ryen W. White. (KDD 2008)
  - HMM + Sequence Mining
- *Fast Logistic Regression for Text Categorization with Variable-Length N-grams*. Georgiana Ifrim, Goekhan Bakir, Gerhard Weikum. (KDD 2008)
  - Logistic regression + Sequence Mining

Why not kernels?

- Kernels take all features into account
  - Graph kernels, Tree kernels, Weighted degree kernels
- The prediction rules are not accountable

- Rescue: Mine the prediction rule of SVM
  - POIMS by S. Sonnenburg et al. (2008)
  - Also meaningful direction to pursue

Three generations of structurization

#iter

- 1G: AdaBoost (2004) — 1000s
- 2G: L1 regularization & Column generation (- 2008) — 100s
  - gBoost, gLARS, iBoost etc..
- 3G: Krylov subspace methods — 10s
  - Partial least squares, Lanczos methods, conjugate gradient

Overview

- Quick Review on Graph Mining
- Graph Partial Least Squares Regression (gPLS) KDD 2008
- Graph Principal Component Analysis (gPCA) ICDM 2008

Quick Review of Graph Mining

Graph Mining

- Analysis of Graph Databases
  - Find all patterns satisfying predetermined conditions
  - Frequent Substructure Mining
- Combinatorial, Exhaustive
- Recently developed
  - AGM (Inokuchi et al., 2000), gspan (Yan et al., 2002), Gaston (2004)

Graph Mining

- Frequent Substructure Mining
  - Enumerate all patterns occurred in at least m graphs

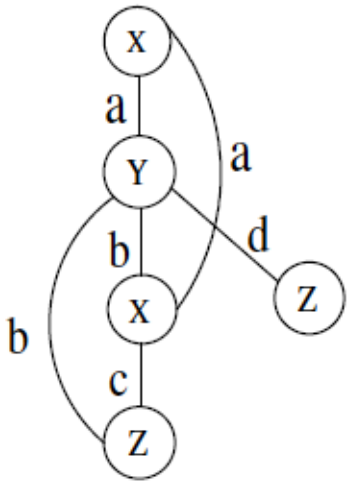
$$S_{freq} = \{k \mid \sum_{i=1}^{n} x_{ik} \geq m\}.$$

$x_{ik} \in \{0, 1\}$:Indicator of pattern k in graph i

Support(k): # of occurrence of pattern k

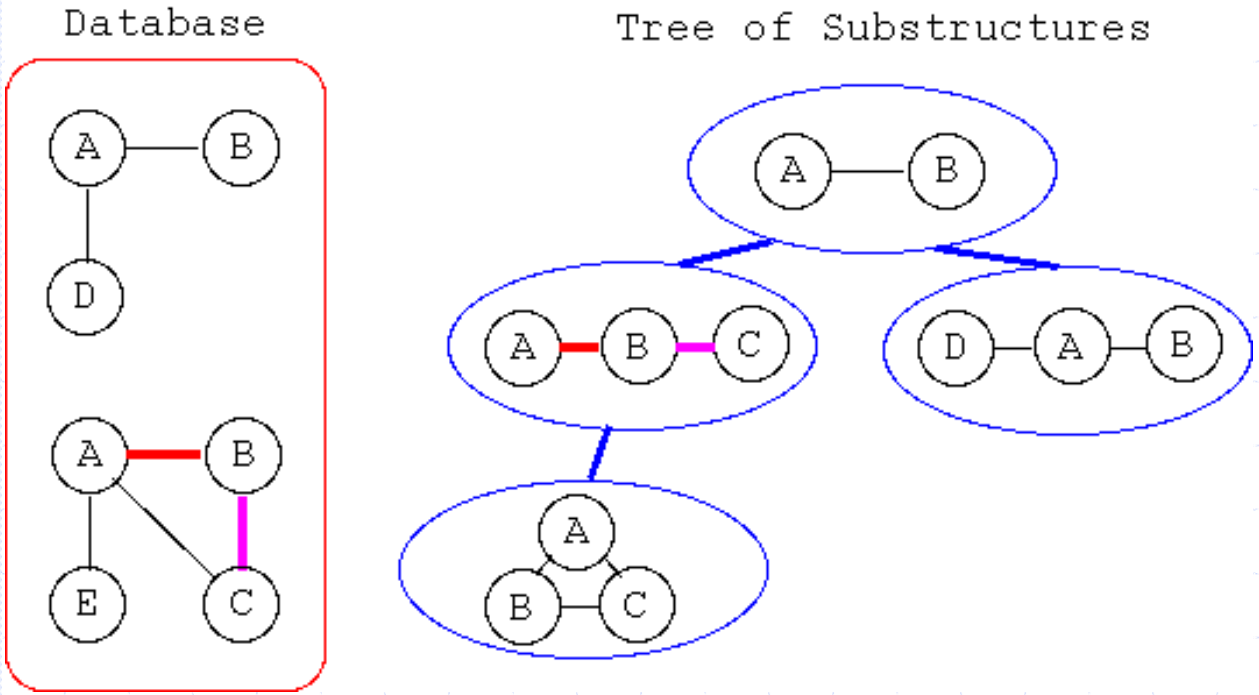
Gspan (Yan and Han, 2002)

- Efficient Frequent Substructure Mining Method
- DFS Code
  - Efficient detection of isomorphic patterns
- Extend Gspan for our works



Enumeration on Tree-shaped Search Space

- Each node has a pattern
- Generate nodes from the root:
  - Add an edge at each step



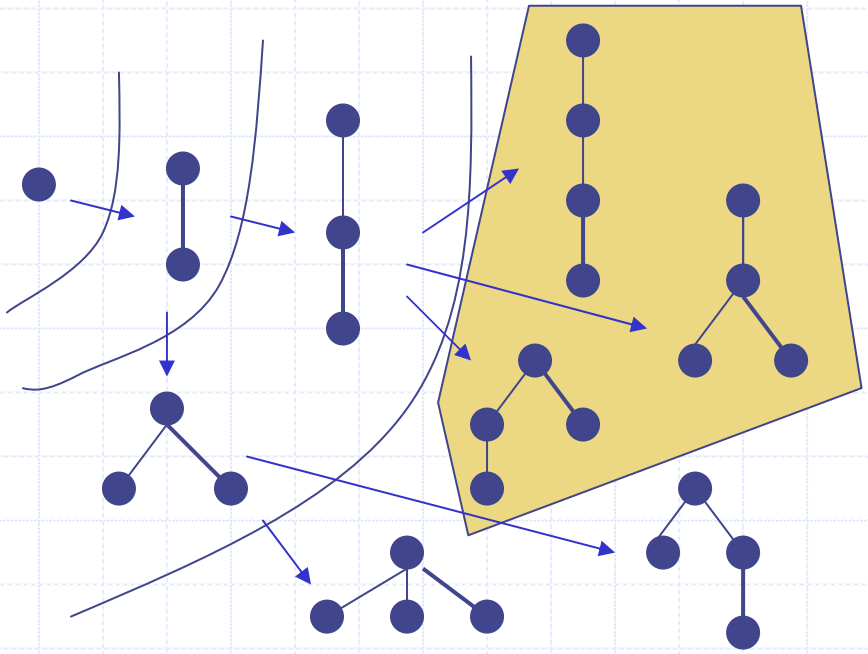
Support(g): # of occurrence of pattern g

Tree Pruning

- Anti-monotonicity:

$$g \subseteq g' \Rightarrow support(g) \geq support(g')$$

- If support(g) < m, stop exploring!



Discriminative patterns: Weighted Substructure Mining

- w_i > 0: positive class
- w_i < 0: negative class
- Weighted Substructure Mining

$$S_w = \{k \mid \left| \sum_{i=1}^{n} w_i(2x_{ik} - 1) \right| \geq \tau\},$$

- Patterns with large frequency difference
- Not Anti-Monotonic: Use a bound for pruning

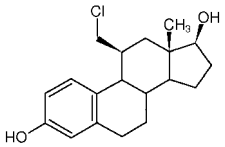
Graph PLS

Problem:

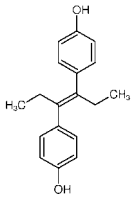
QSAR (Quantitative Structure Activity Relationship)

- Input: Set of chemical compounds
- Output: Activity

Chemical compounds

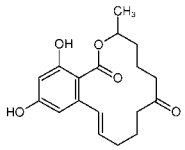


3



1.2

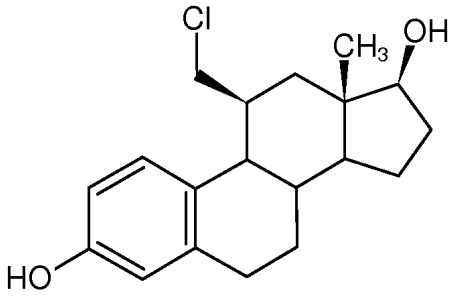
Activity



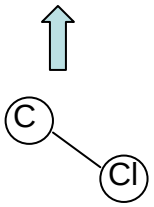
?

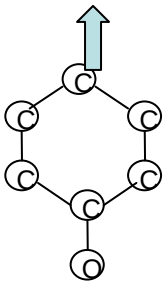
Binary representation of feature vector

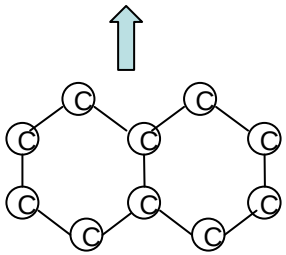
- Binary vector of pattern indicators
- Huge dimensionality
  - Too expensive to mine at once
  - Progressive feature collection



= (1, 0, … 1, 0, … 1, 0, 0, ……)





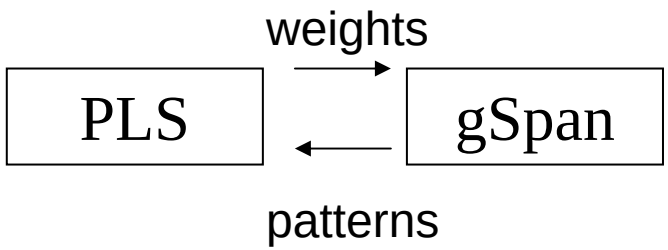


PLS (Partial Least Squares Regression)

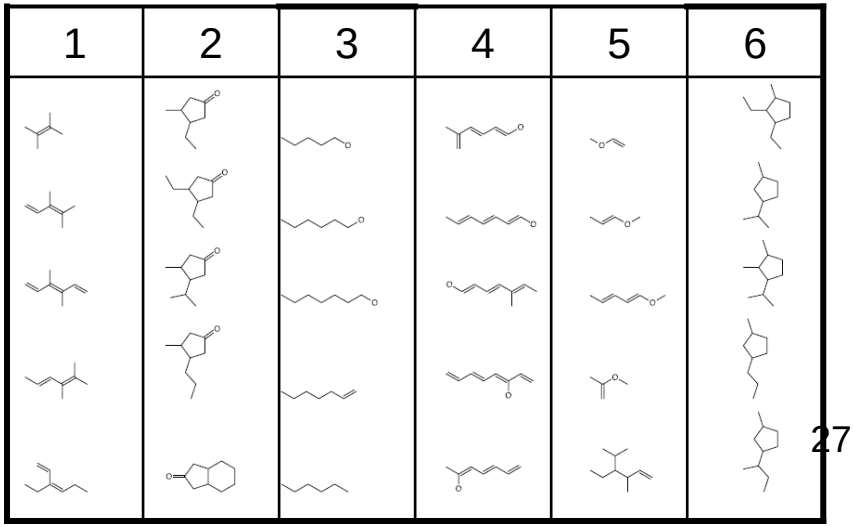
- Design matrix: X
- Response: y
- Dimensionality reduction
  - Weight matrix W： Project examples to a lower dimensional space
  - Latent components T： Projected examples T=WX
- Learn W to maximize the covariance between y and T
- How to perform PLS without accessing all columns of X ?

Graph PLS

- PLS calls graph mining repeatedly
- In i-th iteration,
  - A set of graph patterns corresponding to i-th latent component t_i is collected by weighted subgraph mining



Latent components



PLS algorithm

Algorithm 2 Non-deflation Sparse PLS algorithm.

1: for $i = 1, \ldots, m$ do
2: $r_i = (I - T_{i-1} T_{i-1}^\top) y$ ▷ Residue
3: $v_i = X^\top r_i / \eta.$ ▷ Pre-weight vector
4: $w_i = v_i - \sum_{j=1}^{i-1} (w_j^\top X^\top X v_i) w_j$ ▷ Weight vector
5: $t_i = X w_i$ ▷ Latent components
6: end for

$$\eta = (\boldsymbol{r}_i X X^\top X X^\top \boldsymbol{r}_i)^{1/2}$$

Idea of structurization

- Pre-weight vector computation (step 2)

$$\boldsymbol{v}_i = X^\top \boldsymbol{r}_i / \eta$$

- Collect the features with major pre-weights by weighted substructure mining

$$P_i = \{p \mid \left| \sum_{j=1}^{n} r_{ij} x_{jp} \right| \geq \eta\epsilon\}$$

- Other pre-weights are set to zero

Graph PLS

Algorithm 3 gPLS

1: $r_1 = y,\ X = \emptyset$
2: for $i = 1, \ldots, m$ do
3: $\quad P_i = \{p \mid \left|\sum_{j=1}^{n} r_{ij} x_{jp}\right| \geq \eta\epsilon\}$ ▷ Pattern search
4: $\quad X_{P_i}$: design matrix restricted to P_i
5: $\quad X \leftarrow X \cup X_{P_i}$
6: $\quad v_i = X^\top r_i / \eta$ ▷ Pre-weight vector
7: $\quad w_i = v_i - \sum_{j=1}^{i-1} (w_j^\top X^\top X v_i) w_j$ ▷ Weight vector
8: $\quad t_i = X w_i$ ▷ Latent component
9: $\quad r_{i+1} = r_i - (y^\top t_i) t_i$ ▷ Update residues
10: end for

Classification/regression performance

- 6 chemical compound datasets, 5 fold cross-validation
- Accuracies of gPLS are competitive, but gPLS is substantially faster.
  - Numerical computation much simpler than LP

		gPLS			gBoost		
	data size	Mining Time	Numerical TIme	AUC/Q2	Mining Time	Numerical Time	AUC/Q2
EDKB	59	16.0	0.0025	0.647	15.6	83.3	0.639
CPDB	684	26.8	0.474	0.862	22.8	344	0.862
CAS	437	3579	14.1	0.870	8630	391	0.867
AIDS1	1324	290	0.0652	0.773	783	299	0.752
AIDS2	40939	50300	167	0.747	over 24h		
AIDS3	39965	57100	509	0.883	over 24h		

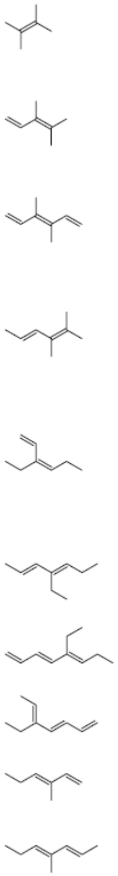
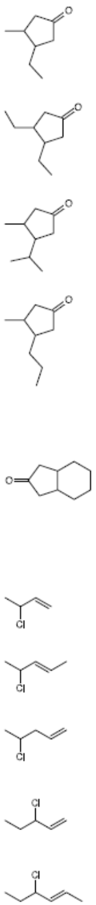
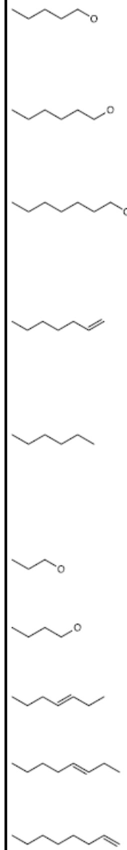
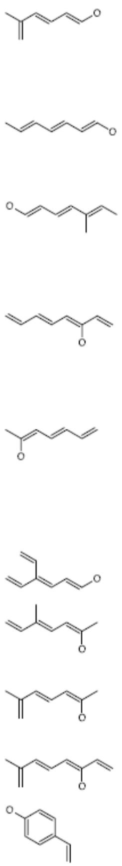
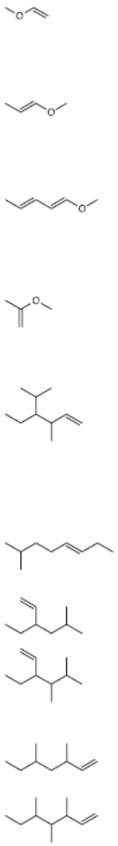
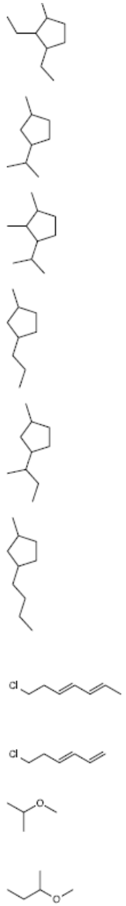
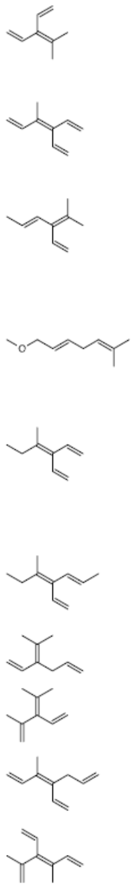
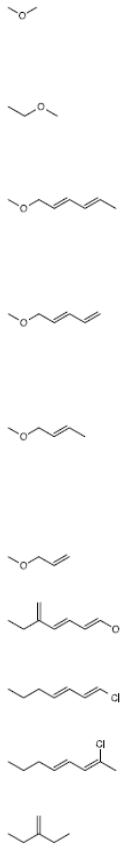
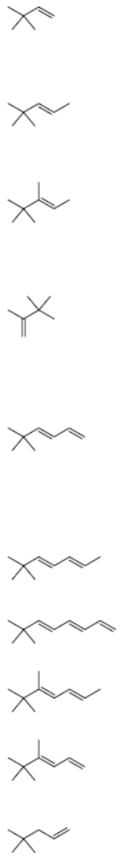
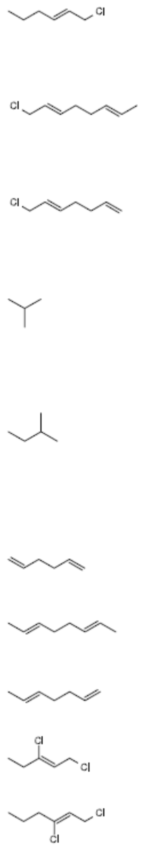
gPLS efficiency

- “Frequent mining + PLS” vs gPLS
- gPLS scales better in larger maximum pattern size threshold (maxpat)

Table 3: Frequent mining + PLS vs gPLS in the CPDB dataset

	frequent mining + PLS				gPLS			
maxpat	# patterns	mining time	numerical time	AUC	# patterns	mining time	numerical time	AUC
1	17	0.0927	0.038	0.696	15.2	0.308	0.038	0.700
2	61	0.148	0.0164	0.770	45.8	1.20	0.1169	0.782
3	182	0.212	0.0335	0.812	73.8	1.09	0.0573	0.833
4	515	0.282	0.0923	0.842	82.6	2.06	0.0488	0.857
5	1387	0.602	0.221	0.846	93.4	1.97	0.0296	0.844
6	3500	2.55	0.525	0.852	85.6	2.67	0.0222	0.833
7	8215	4.38	1.60	0.848	65.4	3.19	0.0146	0.837
8	18107	7.6	5.32	0.840	172	13.1	0.247	0.857
9	37719	17.9	7.42	0.840	209	12.7	0.282	0.859
10	74857	40.3	51.2	0.842	244	26.8	0.474	0.862
11	143006	70.3	92.8	0.835	244	35.4	0.375	0.862
12		out of memory			244	46.3	0.367	0.862
13		out of memory			244	52.4	0.549	0.861
∞		out of memory			244	66.3	0.586	0.861

Latent components by gPLS

1	2	3	4	5	6	7	8	9	10
									

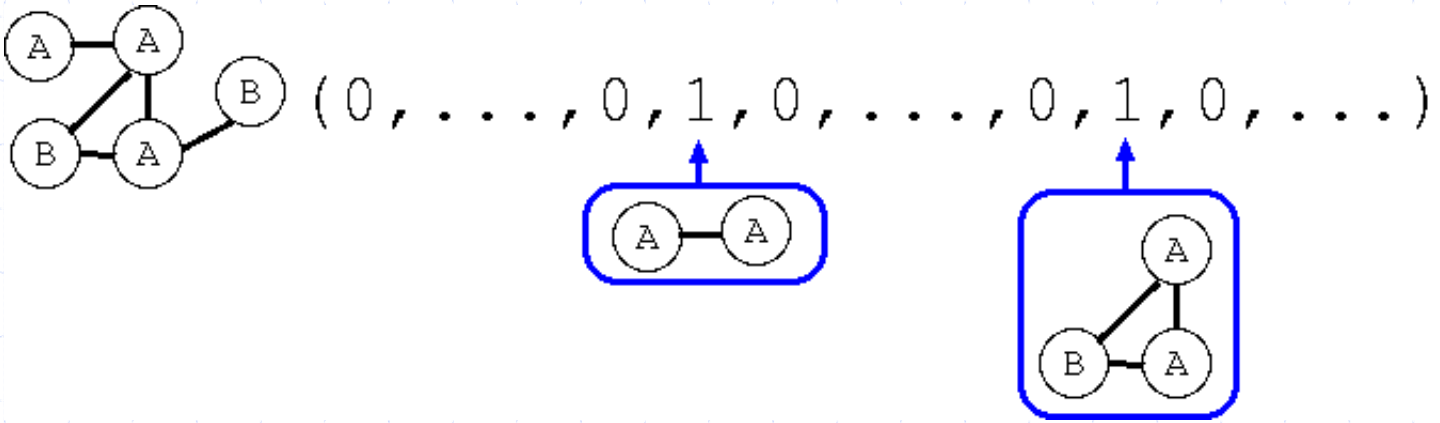
Summary (gPLS)

- Introduction of a new PLS algorithm for use with graph mining.
- High efficiency compared with the naïve counterpart.
- Graph mining algorithm can be replaced with other frequent pattern mining algorithms such as itemset mining, sequence mining and tree mining.

Graph PCA

Graph PCA method

- Design matrix X
- Gram matrix $A = XX^\top$
- Derive eigenvalue and eigenvector of A
  - Equivalent to kernel PCA with linear kernel
- How to perform eigendecomposition without accessing all columns of X ?



Lanczos algorithm

$$T = \begin{bmatrix} \alpha_1 & \beta_1 & \cdots & 0 \\ \beta_1 & \alpha_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \beta_{n-1} \\ 0 & \cdots & \beta_{n-1} & \alpha_n \end{bmatrix}.$$

- Find an orthonormal matrix Q such that

$$T = Q^\top A Q$$

  is a tridiagonal matrix
- Q =(q_1,…,q_m): Lanczos vectors
- Then, T is easily eigendecomposed
  - Eigenvalues preserved in T
  - Eigenvectors of A are recovered as

$$V = QR$$

Eigenvectors of A

Eigenvectors of T

Lanczos method for computing all eigenvectors

Algorithm 1 The Lanczos algorithm.

1: Input: $\boldsymbol{A} \in \mathbb{R}^{n \times n}, \boldsymbol{q}_1 \in \mathbb{R}^n, \|\boldsymbol{q}_1\| = 1.$
2: Output: All eigenpairs $[\boldsymbol{\lambda}, V]$
3: Initial: $\boldsymbol{r}_0 = \boldsymbol{q}_1;\ \beta_0 = 1;\ k = 0$
4: **while** $\beta_k \neq 0$ **do**
5: $\quad \boldsymbol{q}_k = \boldsymbol{r}_k / \beta_k$
6: $\quad k = k + 1$
7: $\quad \alpha_k = \boldsymbol{q}_k^\top \boldsymbol{A} \boldsymbol{q}_k$
8: $\quad \boldsymbol{r}_k = \boldsymbol{A}\boldsymbol{q}_k - \beta_{k-1}\boldsymbol{q}_{k-1} - \alpha_k \boldsymbol{q}_k$
9: $\quad \beta_k = \|\boldsymbol{r}_k\|_2$
10: **end while**
11: $[\boldsymbol{\lambda}, R] = EigenDecomposition(\boldsymbol{T})$
12: $V = QR$

Importance of initial Lanczos vector q_1

- If q_1 corresponds to the first eigenvector, Lanczos finishes in n iterations, creating all eigenvectors
- Typically, the first vector is not known, so more iterations are necessary

- If one needs only major eigenvectors, early stopping is possible
  - Major eigenvalues converge faster
  - Number of iterations determined by convergence bound (Saad, 1980)

Idea of structurization

- Substitute $A = XX^\top$
- Lanczos accesses X only through $\boldsymbol{g}_k = X^\top \boldsymbol{q}_k$

$$g_{ki} = \sum_{i=1}^{n} x_{ip} q_{kj}.$$

- Collect the features whose $|g_{ki}|$ are large

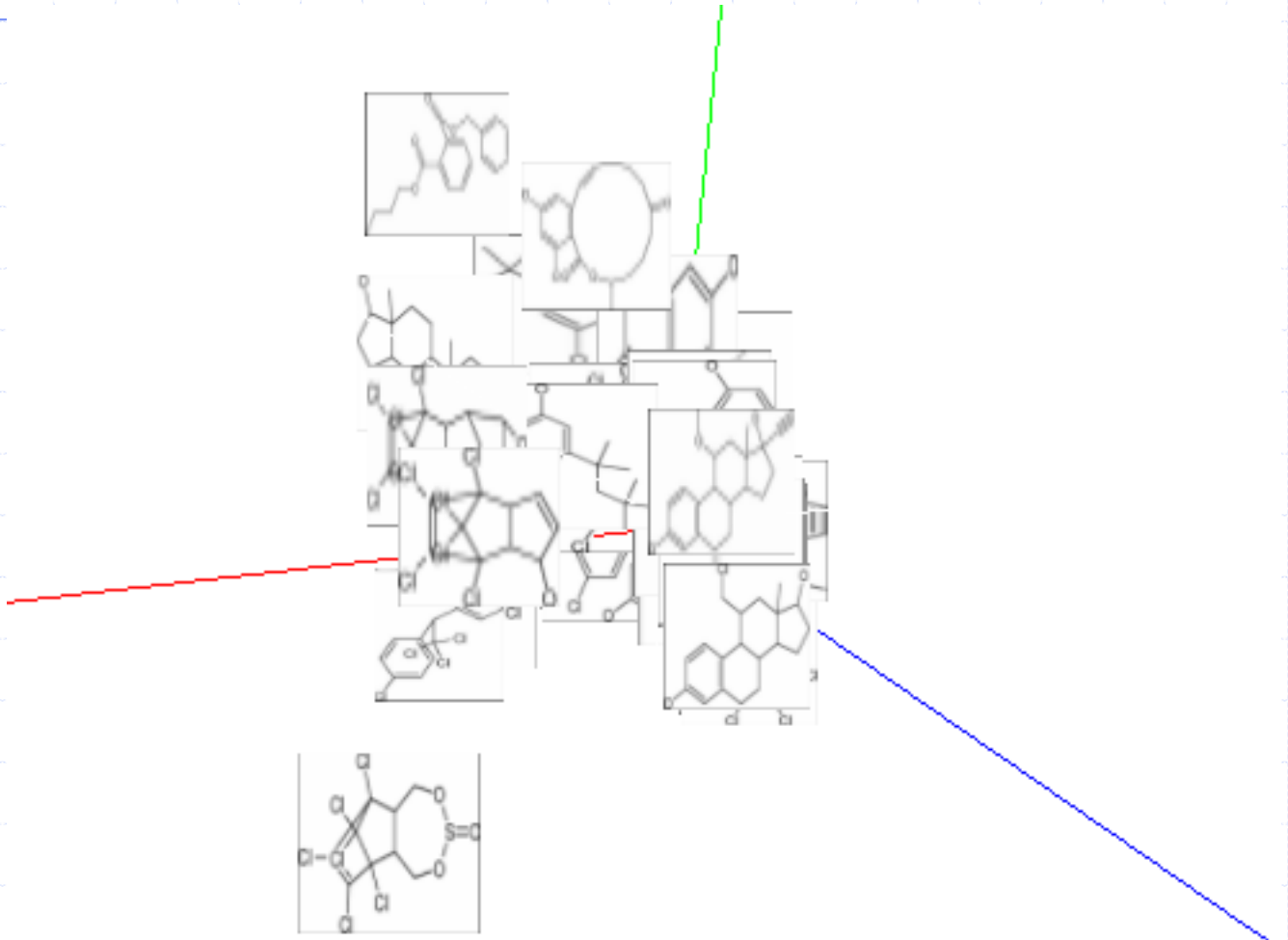
$$\{i \mid |g_{ki}| \geq \epsilon\}$$

- Implemented by weighted subgraph mining

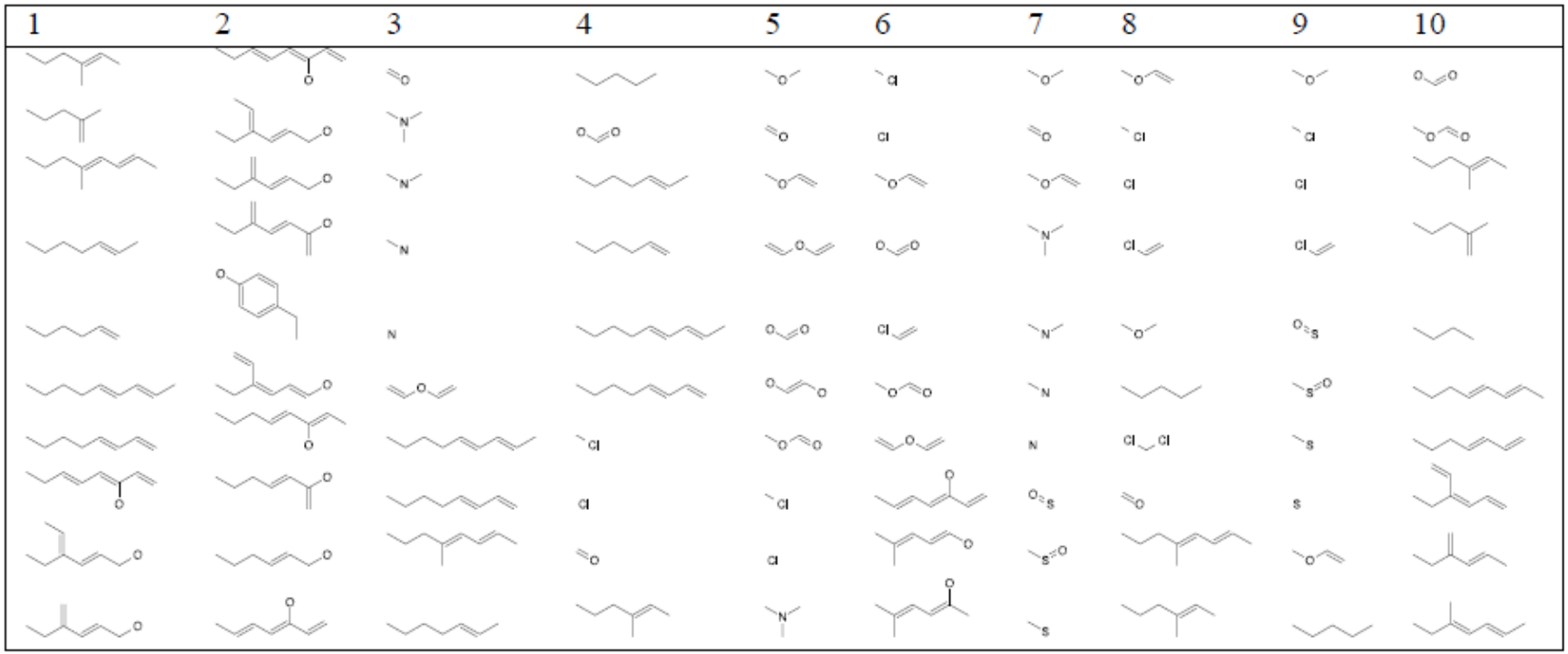
Algorithm 4 Graph PCA

Input: Graphs $G_1 \ldots, G_n$, Tolerance ϵ
Output: Patterns P, Principal components $Z \in \Re^{|P| \times m}$
Initial: $\boldsymbol{q}_1 = \mathbf{1}/\sqrt{n};\ \boldsymbol{r}_0 = \boldsymbol{q}_1;\ \beta_0 = 1;\ k = 0;\ P = \emptyset$
while (4) is not true **do**
    $\boldsymbol{q}_k = \boldsymbol{r}_k / \beta_k$
    $k = k + 1$
    $P_k = \{p \mid \left| \sum_{j=1}^{n} q_{kj} x_{jp} \right| \geq \epsilon\}$ ▷ Pattern search
    $P \leftarrow P \cup P_k$
    X_P: design matrix restricted to P
    $\alpha_k = \boldsymbol{q}_k^\top X_P X_P^\top \boldsymbol{q}_k$
    $\boldsymbol{r}_k = X_P X_P^\top \boldsymbol{q}_k - \beta_{k-1} \boldsymbol{q}_{k-1} - \alpha_k \boldsymbol{q}_k$
    for $j = 1 : k - 1$ **do**
        $\boldsymbol{r}_k = \boldsymbol{r}_k - \boldsymbol{q}_j (\boldsymbol{q}_j^\top \boldsymbol{r}_k)$
    end for
    $\beta_k = \|\boldsymbol{r}_k\|$
    $[\boldsymbol{\lambda}_k, R_k] = EigenDecomposition(\boldsymbol{T}_k)$
    $V_k = Q_k R_k$
end while
Truncate $[\boldsymbol{\lambda}_k, V_k]$ to top m eigenpairs
$Z = X_P^\top V_m$

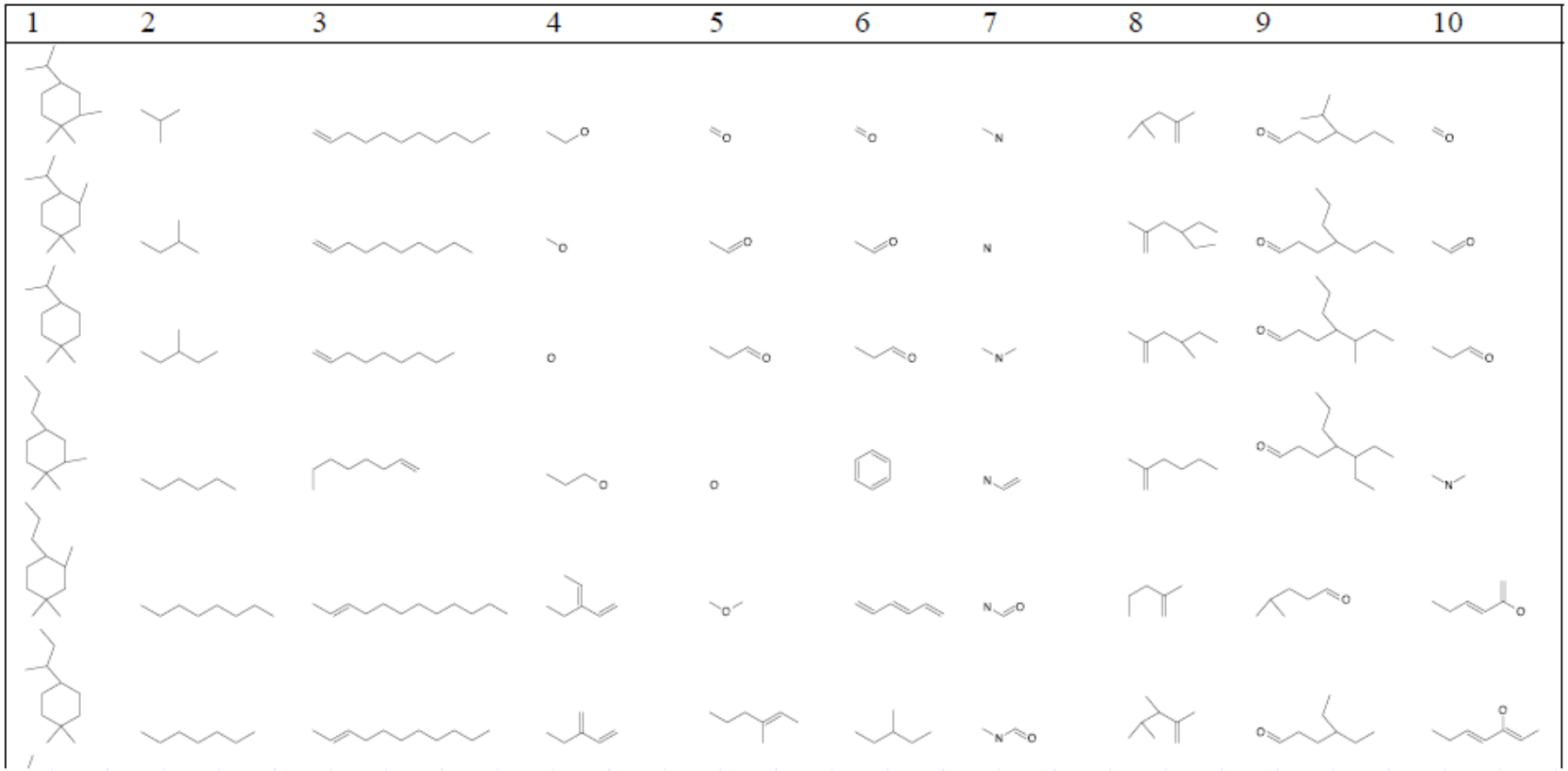
Three dimensional plot of EDKB-ER



Principal Components of EDKB-ER



Principal Components of EDKB-AR



Summary (gPCA)

- In gPCA, patterns are summarized in several components
- Each component characterize different aspects of graph database
- Easier to browse than frequent patterns

Conclusion

- Combination of mining and learning algorithms result in a very powerful framework
- Interpretability is the key
- Successful applications to biological and computer vision domains
- Further applications possible to many different fields

Publication and software

H. Saigo et al., KDD 2008
H. Saigo and K. Tsuda, ICDM 2008
H. Saigo et al., Machine Learning, to appear
H. Saigo et al., Bioinformatics, 2007
S. Nowozin et al., CVPR 2007
S. Nowozin et al., ICCV 2007
K. Tsuda, ICML 2006, ICML 2007

iboost: itemset boosting
http://www.kyb.mpg.de/bs/people/hiroto/iboost/

gboost: graph boosting
http://www.kyb.mpg.de/bs/people/nowozin/gboost/

pboost: sequence boosting
http://www.kyb.mpg.de/bs/people/nowozin/pboost/