Machine Learning and Data Mining: Challenges and Opportunities for CP

Luc De Raedt and Siegfried Nijssen

KATHOLIEKE UNIVERSITEIT LEUVEN

CP 2011

Dagstuhl Workshop
CP meets DM/ML

SCIENTIFIC

ADVISORY

MACHINE LEARNING &
DATA MINING
PERSPECTIVE

SCIENTIFIC

ADVISORY

MACHINE LEARNING &
DATA MINING
PERSPECTIVE

WARNING

TUTORIAL IS
INCOMPLETE WRT
STATE OF ART

WE PRESENT A FLAVOR
OF TECHNIQUES THAT
WE FEEL ARE USEFUL

LOGIC-BASED

Questions

1. Can CP problems and CP solvers help to formulate and solve ML / DM problems ?
2. Can ML and DM help to formulate and solve constraint satisfaction problems ?

We shall argue that the answer to both questions is YES
At the same time, we shall introduce some ML/DM techniques as well as some challenges and opportunities

The CP perspective

Formulating the model is a knowledge acquisition task

Improving the performance of solvers is speed-up learning

Machine learning may help as shown by several initial works

The ML/DM Perspective

Machine Learning is a (constrained) optimization problem

- learning functions

Data mining is often constraint satisfaction

- “Constraint based mining”

Still ML/DM do not really use CP ...

Constraint-Based Mining

Numerous constraints have been used

Numerous systems have been developed

And yet,

- new constraints often require new implementations
- very hard to combine different constraints

Constraint Programming

Exists since about 20 ? years

A general and generic methodology for dealing with constraints across different domains

Efficient, extendable general-purpose systems exist, and key principles have been identified

Surprisingly CP has not been used for data mining ?

CP systems often more elegant, more flexible and more efficient than special purpose systems

Also true for Data Mining ?

Overview

How CP can be used in ML / DM (Siegfried)

- introduction to constraint-based mining
- introduction to constraint-clustering
- challenges

How ML might help CP (Luc)

- learning the model from data
- introduction to some ML techniques

How CP can help DM

Constraints in Data Mining

- Pattern Mining
- Decision Trees
- Clustering

Pattern Mining

- Basic setting: frequent itemset mining
  - Data miner's solution
  - Constraint programming solution
- Extensions
  - Constraint-based mining
    - Common constraints
    - Constraint programming solution
  - Other types of data
  - Pattern set mining

Frequent Itemset Mining

support()=3

- Market basket data

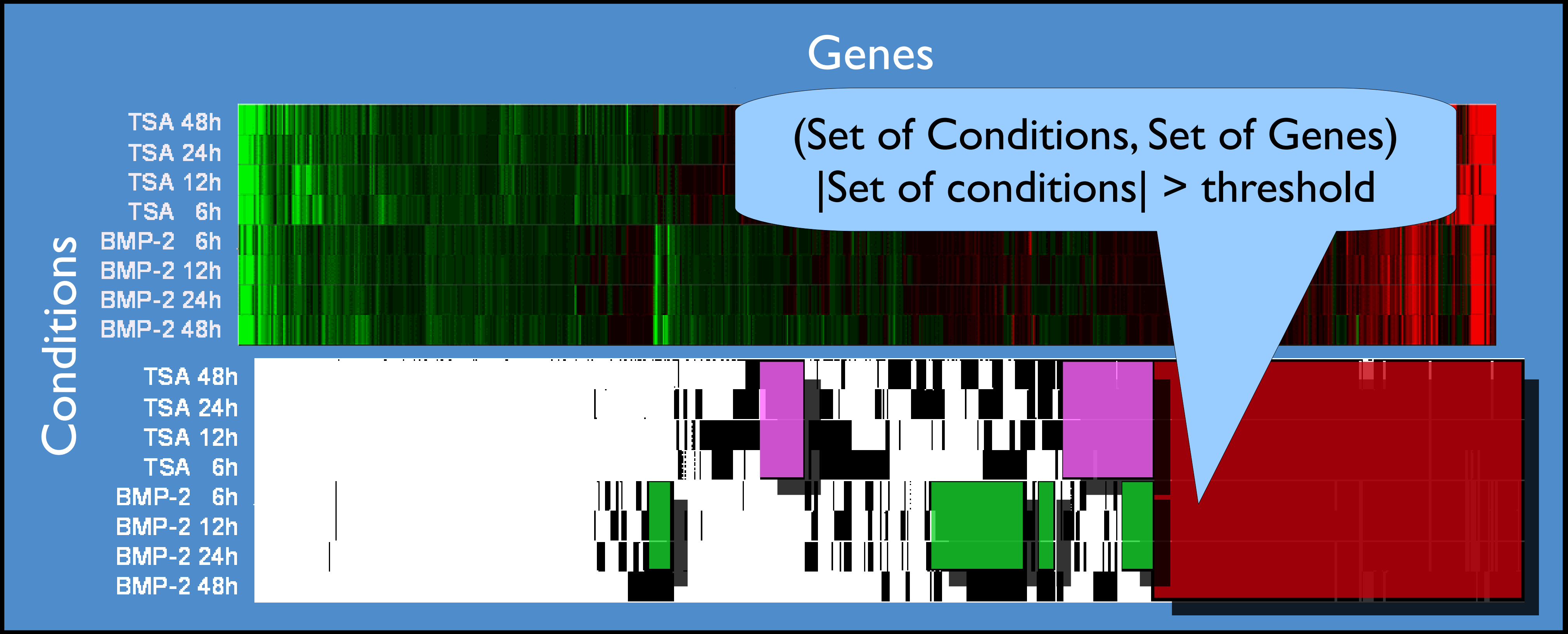
Frequent Itemset Mining

- **Given**
  - A database with sets of items
  - A support threshold
- **Find**
  - **ALL** subsets of items *I* for which support(*I*)>threshold

[Agrawal 1996]

Frequent Itemset Mining

- Gene expression data



Frequent Itemset Mining

```
4.9,3.1,1.5,0.1,Iris-setosa
5.0,3.2,1.2,0.2,Iris-setosa
5.5,3.5,1.3,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
4.4,3.0,1.3,0.2,Iris-setosa
5.1,3.4,1.5,0.2,Iris-setosa
5.0,3.5,1.3,0.3,Iris-setosa
4.5,2.3,1.3,0.3,Iris-setosa
4.4,3.2,1.3,0.2,Iris-setosa
5.0,3.5,1.6,0.6,Iris-setosa
5.1,3.8,1.9,0.4,Iris-setosa
4.8,3.0,1.4,0.3,Iris-setosa
5.1,3.8,1.6,0.2,Iris-setosa
4.6,3.2,1.4,0.2,Iris-setosa
5.3,3.7,1.5,0.2,Iris-setosa
5.0,3.3,1.4,0.2,Iris-setosa
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
```

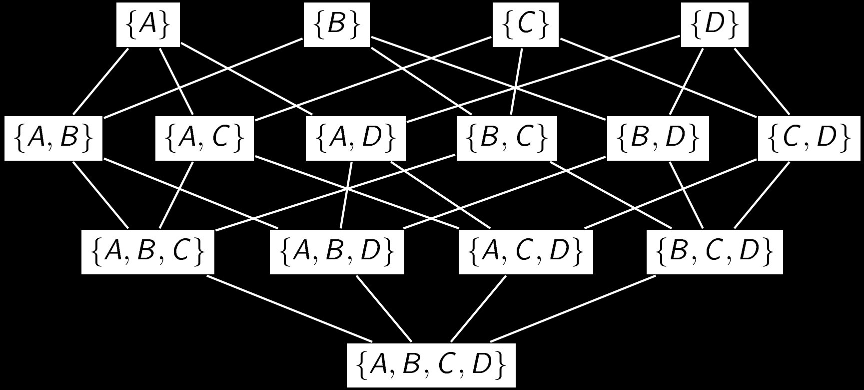
if Petal length >= 2.0 and
Petal width <= 0.5
then Iris-Setosa
else Iris-Versicolor

Item

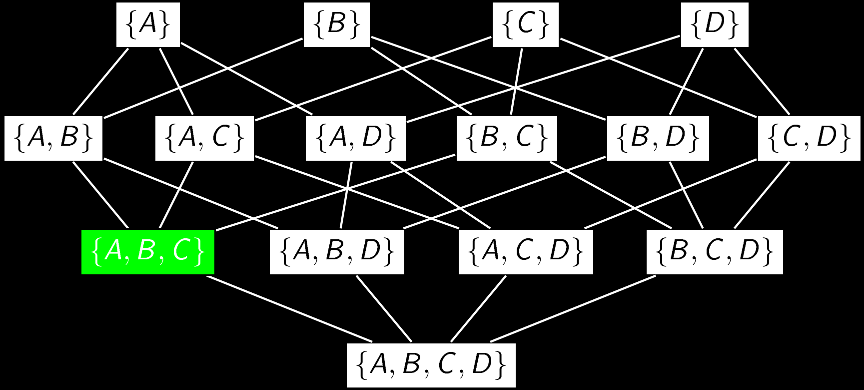
Frequent Itemset Mining

- Algorithms
  - Pruning based on “anti-monotonicity”
  - Many different search orders
    - Breadth-first
    - Depth-first
  - Many different data structures
    - How to store lots of data in memory during the search?

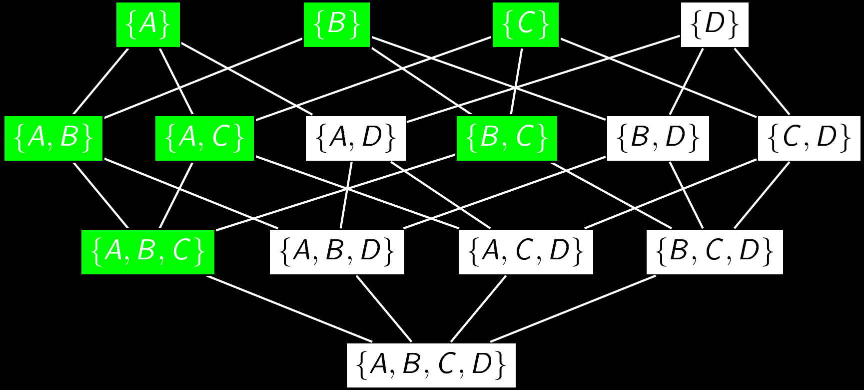
Anti-monotonocity



Anti-monotonocity

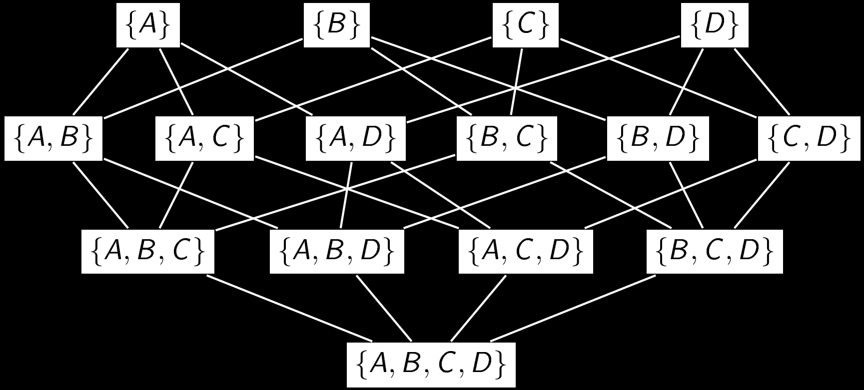


Anti-monotonocity

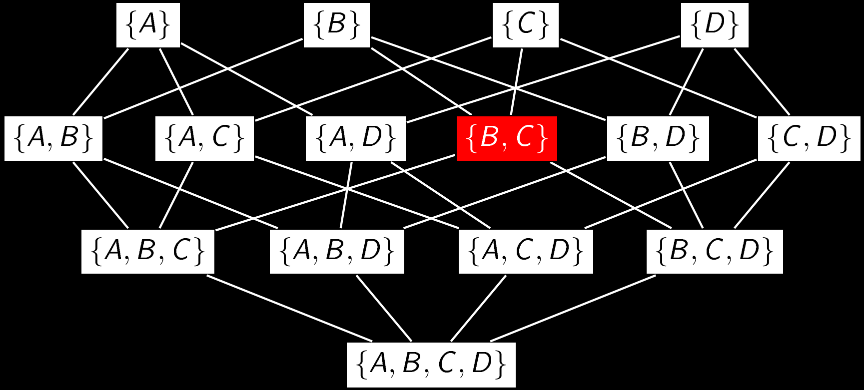


- Anti-monotonicity: subsets of frequent itemsets are frequent

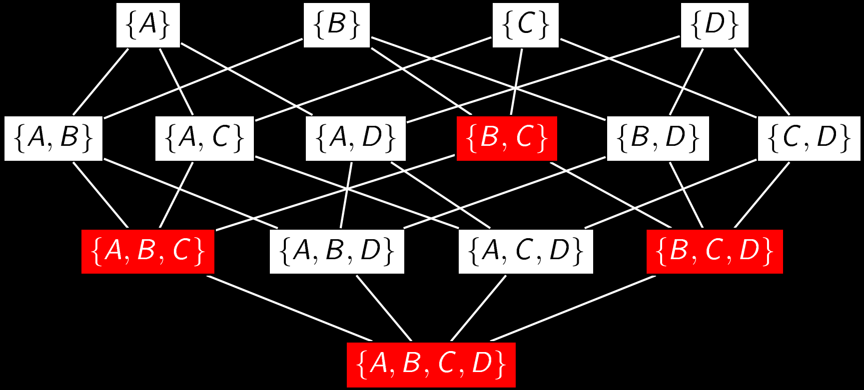
Anti-monotonicity



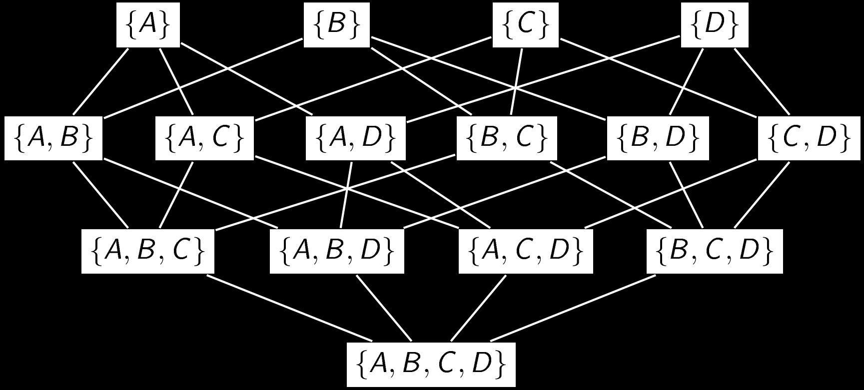
Anti-monotonicity



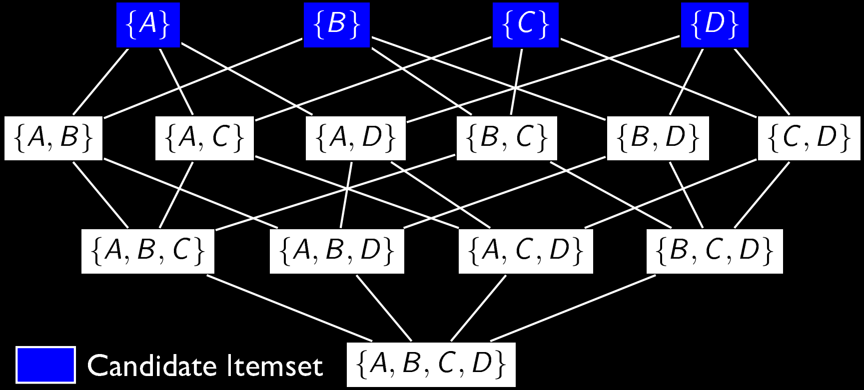
Anti-monotonicity



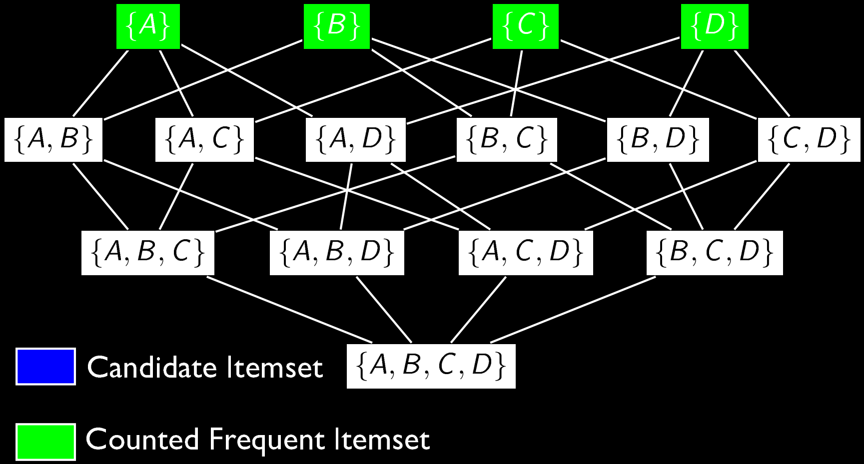
Search: Apriori



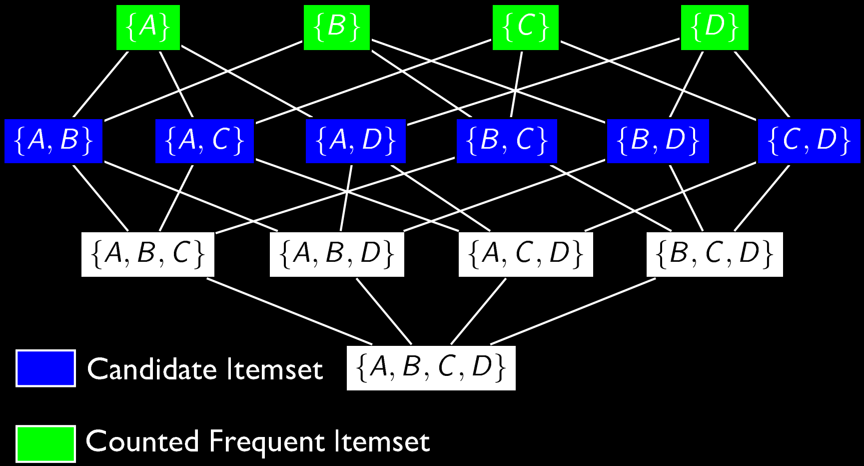
Search: Apriori



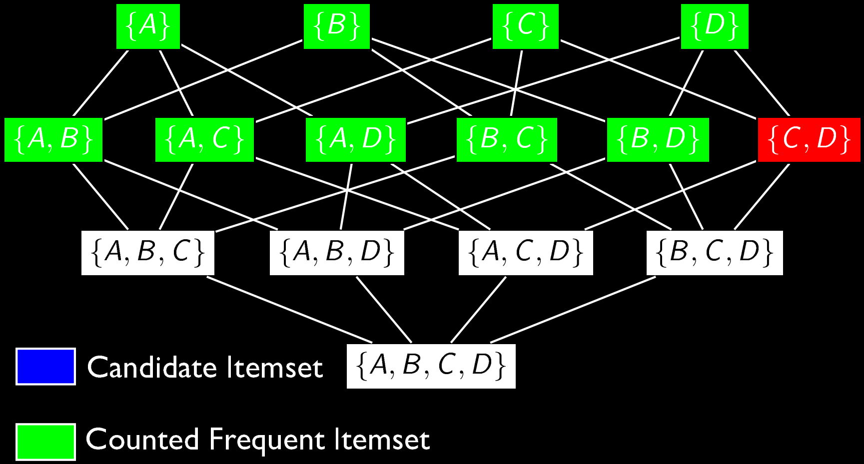
Search: Apriori



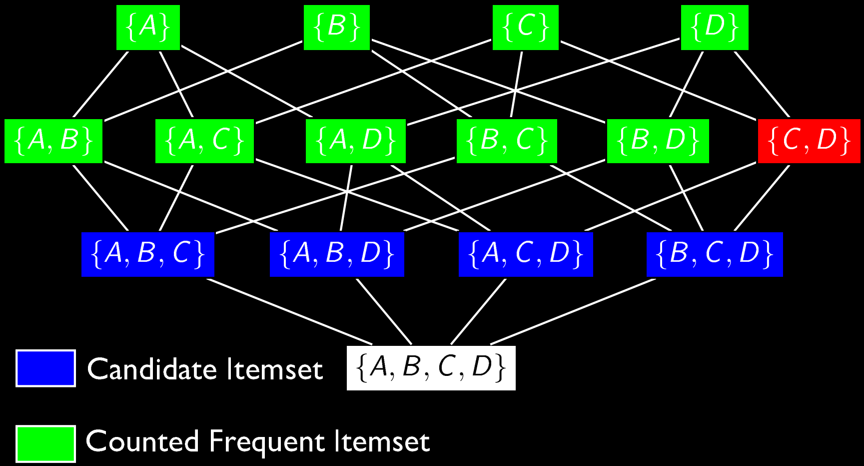
Search: Apriori



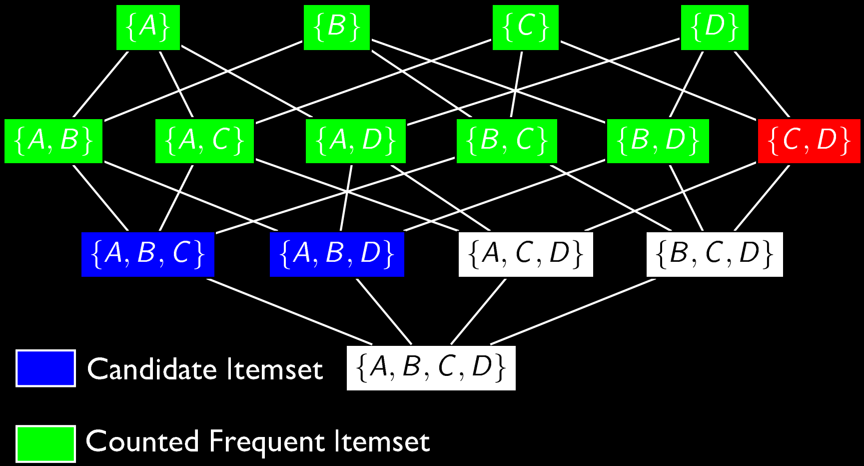
Search: Apriori



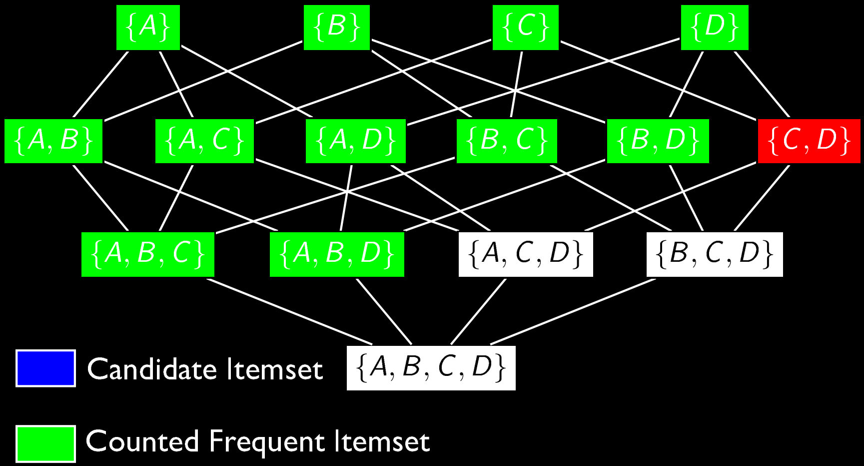
Search: Apriori



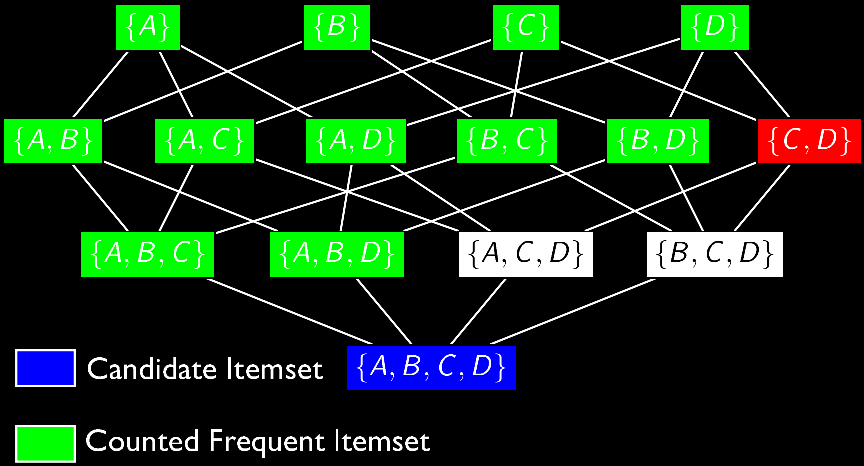
Search: Apriori



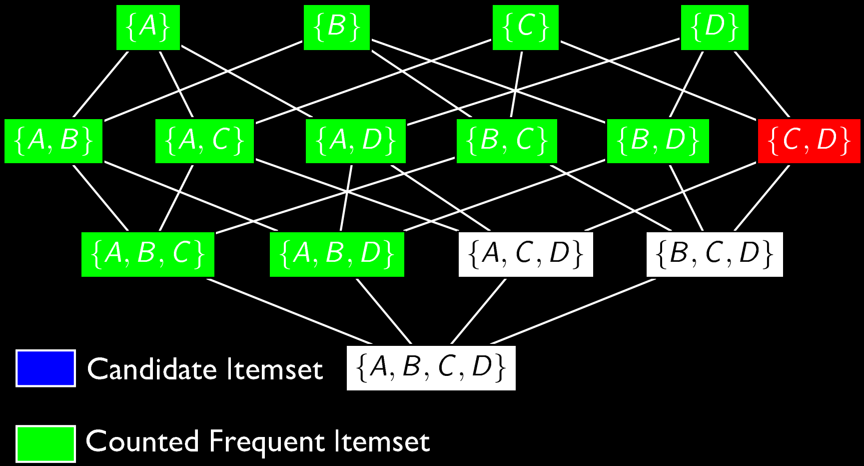
Search: Apriori



Search: Apriori



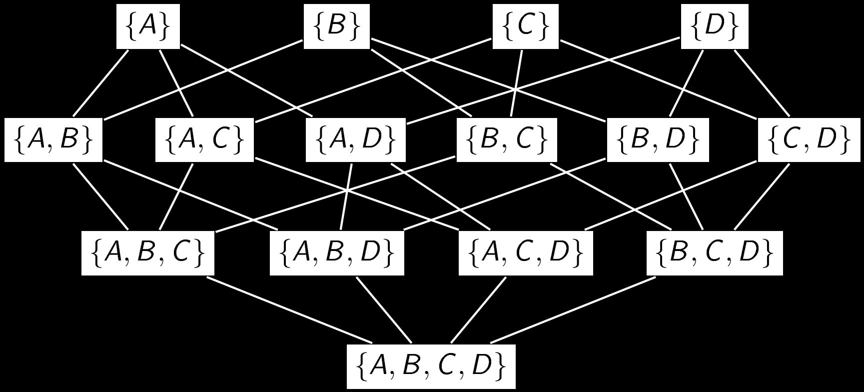
Search: Apriori



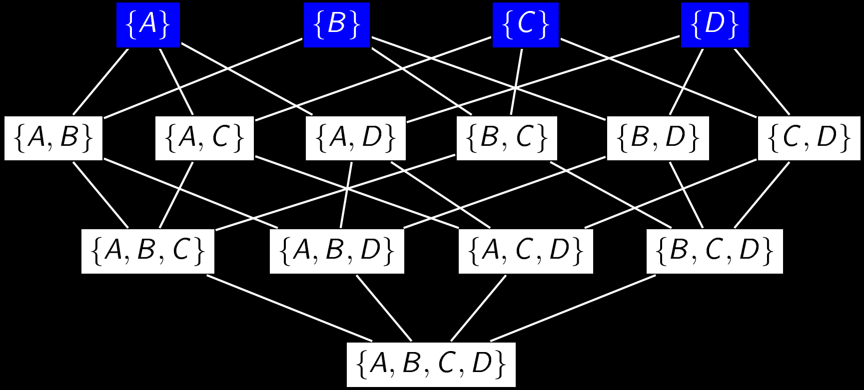
Search: Apriori

- Benefits:
  - Limited number of passes when the database is on disk
  - Maximal pruning before counting

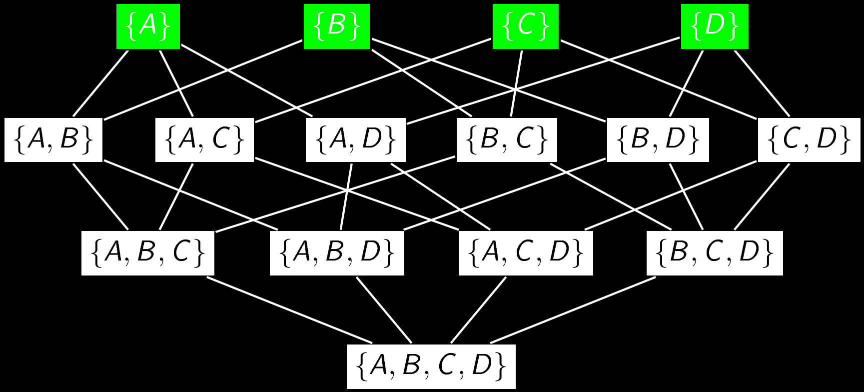
Depth-First Search



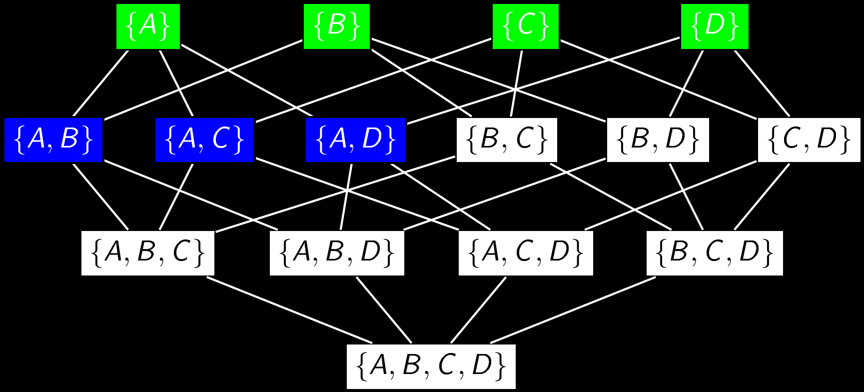
Depth-First Search



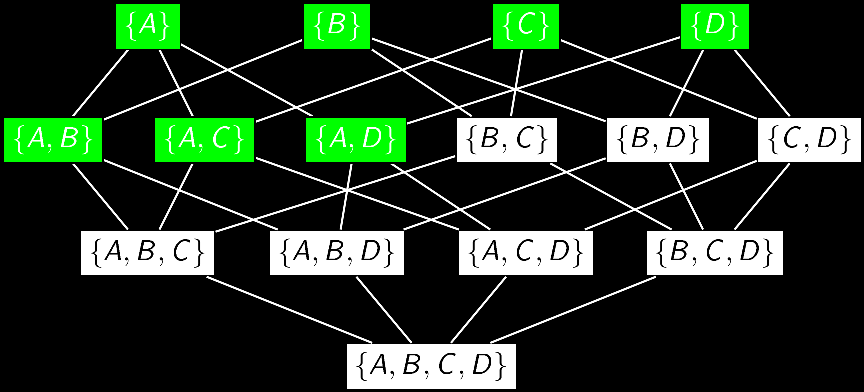
Depth-First Search



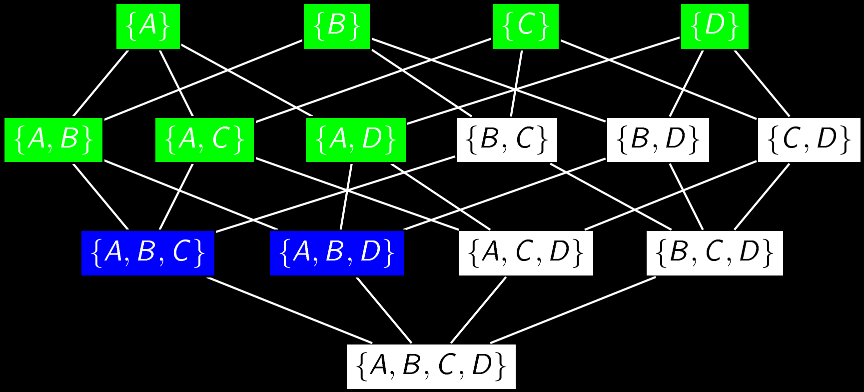
Depth-First Search



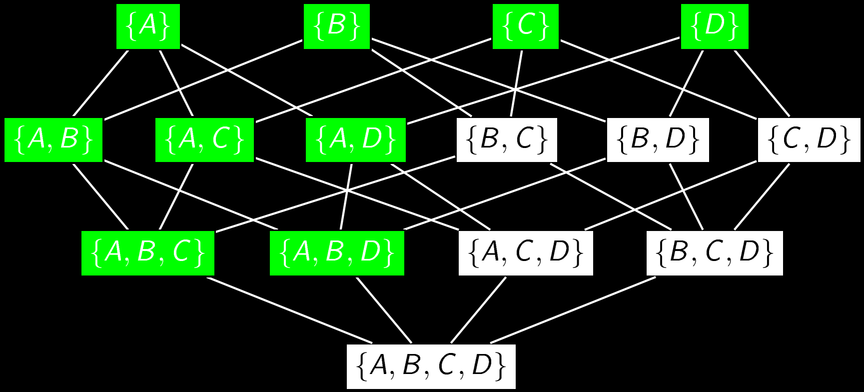
Depth-First Search



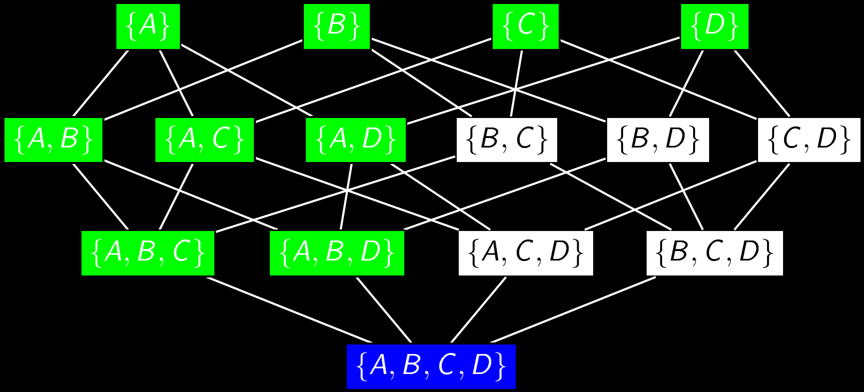
Depth-First Search



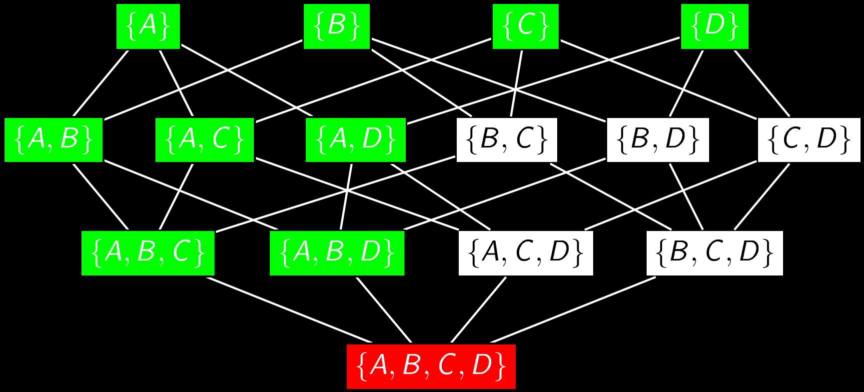
Depth-First Search



Depth-First Search



Depth-First Search



Depth-First Search

- Benefits:
  - Less candidates at the same time in main memory $\Rightarrow$ memory can be used for other purposes
  - More efficient in practice

Frequent Itemset Mining in CP

- variables

  $[I_1 \ldots I_n], [T_1 \ldots T_m]$

- domains

  $I_x, T_y = \{0, 1\}$

- constraints
  - support

$$\sum_t T_t \geq minsup$$

	$[I_1$	I_2	I_3	I_4	I_5	I_6	I_7	$I_8]$
T_1	0	1	1	1	0	0	0	0
T_2	0	0	0	0	0	0	1	1
T_3	1	0	0	0	1	1	0	0
T_4	1	1	1	0	0	0	0	0
T_5	1	0	0	1	0	1	0	1
T_6	0	1	0	0	1	0	1	1
T_7	1	0	0	1	0	0	1	1
T_8	1	1	1	0	0	1	0	1
T_9	1	1	0	0	0	1	1	0
T_{10}	1	1	1	1	0	0	0	0
T_{11}	1	0	0	0	0	1	1	1
T_{12}	1	1	1	0	1	1	0	0

[De Raedt et al. 2008]

Frequent Itemset Mining in CP

- variables

  $[I_1 \ldots I_n], [T_1 \ldots T_m]$

- domains

  $I_x, T_y = \{0, 1\}$

- constraints
  - support

$$\sum_t T_t \geq minsup$$

	$[I_1$	I_2	I_3	I_4	I_5	I_6	I_7	$I_8]$
T_1 1)	0	1	1	1	0	0	0	0
T_2 2)	0	0	0	0	0	0	1	1
T_3 3)	1	0	0	0	1	1	0	0
T_4 4)	1	1	1	0	0	0	0	0
T_5 5)	1	0	0	1	0	1	0	1
T_6 6)	0	1	0	0	1	0	1	1
T_7 7)	1	0	0	1	0	0	1	1
T_8 8)	1	1	1	0	0	1	0	1
T_9 9)	1	1	0	0	0	1	1	0
T_{10} 10)	1	1	1	1	0	0	0	0
T_{11} 11)	1	0	0	0	0	1	1	1
T_{12} 12)	1	1	1	0	1	1	0	0

or reified: $I_i = 1 \Rightarrow \sum_t D_{ti} T_t \geq minsup$

Frequent Itemset Mining in CP

- variables

  $[I_1 \dots I_n], [T_1 \dots T_m]$

- domains

  $I_x, T_y = \{0, 1\}$

- constraints
  - support

$$\sum_t T_t \geq minsup$$

or reified: $I_i = 1 \Rightarrow \sum_t D_{ti} T_t \geq minsup$

  - coverage

$$T_t = 1 \Leftrightarrow \sum_i I_i (1 - D_{ti}) = 0$$

		[I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8]
T_1	1)	0	1	1	1	0	0	0	0
T_2	2)	0	0	0	0	0	0	1	1
T_3	3)	1	0	0	0	1	1	0	0
T_4	4)	1	1	1	0	0	0	0	0
T_5	5)	1	0	0	1	0	1	0	1
T_6	6)	0	1	0	0	1	0	1	1
T_7	7)	1	0	0	1	0	0	1	1
T_8	8)	1	1	1	0	0	1	0	1
T_9	9)	1	1	0	0	0	1	1	0
T_{10}	10)	1	1	1	1	0	0	0	0
T_{11}	11)	1	0	0	0	0	1	1	1
T_{12}	12)	1	1	1	0	1	1	0	0

Frequent Itemset Mining in CP

- A transaction is covered iff $I \subseteq D_t$

Frequent Itemset Mining in CP

- A transaction is covered iff $I \subseteq D_t$

$$T_t = 1 \quad \Leftrightarrow \quad I \subseteq D_t$$

Frequent Itemset Mining in CP

- A transaction is covered iff $I \subseteq D_t$

$$T_t = 1 \quad \Leftrightarrow \quad I \subseteq D_t$$
$$T_t = 1 \quad \Leftrightarrow \quad \forall i \in \mathcal{I} : I_i = 1 \rightarrow D_{ti} = 1$$

Frequent Itemset Mining in CP

- A transaction is covered iff $I \subseteq D_t$

$$
\begin{aligned}
T_t = 1 &\Leftrightarrow I \subseteq D_t \\
T_t = 1 &\Leftrightarrow \forall i \in \mathcal{I} : I_i = 1 \rightarrow D_{ti} = 1 \\
T_t = 1 &\Leftrightarrow \forall i \in \mathcal{I} : I_i = 0 \vee D_{ti} = 1
\end{aligned}
$$

Frequent Itemset Mining in CP

- A transaction is covered iff $I \subseteq D_t$

$$
\begin{aligned}
T_t = 1 &\Leftrightarrow I \subseteq D_t \\
T_t = 1 &\Leftrightarrow \forall i \in \mathcal{I} : I_i = 1 \rightarrow D_{ti} = 1 \\
T_t = 1 &\Leftrightarrow \forall i \in \mathcal{I} : I_i = 0 \vee D_{ti} = 1 \\
T_t = 1 &\Leftrightarrow \forall i \in \mathcal{I} : I_i = 0 \vee 1 - D_{ti} = 0
\end{aligned}
$$

Frequent Itemset Mining in CP

- A transaction is covered iff $I \subseteq D_t$

$$
\begin{aligned}
T_t = 1 &\Leftrightarrow I \subseteq D_t \\
T_t = 1 &\Leftrightarrow \forall i \in \mathcal{I} : I_i = 1 \rightarrow D_{ti} = 1 \\
T_t = 1 &\Leftrightarrow \forall i \in \mathcal{I} : I_i = 0 \vee D_{ti} = 1 \\
T_t = 1 &\Leftrightarrow \forall i \in \mathcal{I} : I_i = 0 \vee 1 - D_{ti} = 0 \\
T_t = 1 &\Leftrightarrow \sum_{i \in \mathcal{I}} I_i (1 - D_{ti}) = 0
\end{aligned}
$$

Frequent Itemset Mining in CP

- Model in Minizinc

```
int: NrI; int: NrT;
array [1..NrT,1..NrI] of bool: TDB;
int: Freq;

array [1..NrI] of var bool: Items;
array [1..NrT] of var bool: Trans;

constraint % coverage
   forall(t in 1..NrT) (
     Trans[t] <-> sum(i in 1..NrI) (bool2int(TDB[t,i] → Items[i])) <= 0          );
constraint % frequency
   forall(i in 1..NrI) (
      Items[i] -> sum(t in 1..NrT) (bool2int(TDB[t,i] /\ Trans[t])) >= Freq);

solve satisfy;
```

Search

freq >= 2: $I_i = 1 \Rightarrow \sum_t D_{ti} T_t \geq minsup$

coverage: $T_t = 1 \Leftrightarrow \sum_i I_i (1 - D_{ti}) = 0$

- propagate i2 (freq)
  Intuition: infrequent i2 can never be part of freq. superset

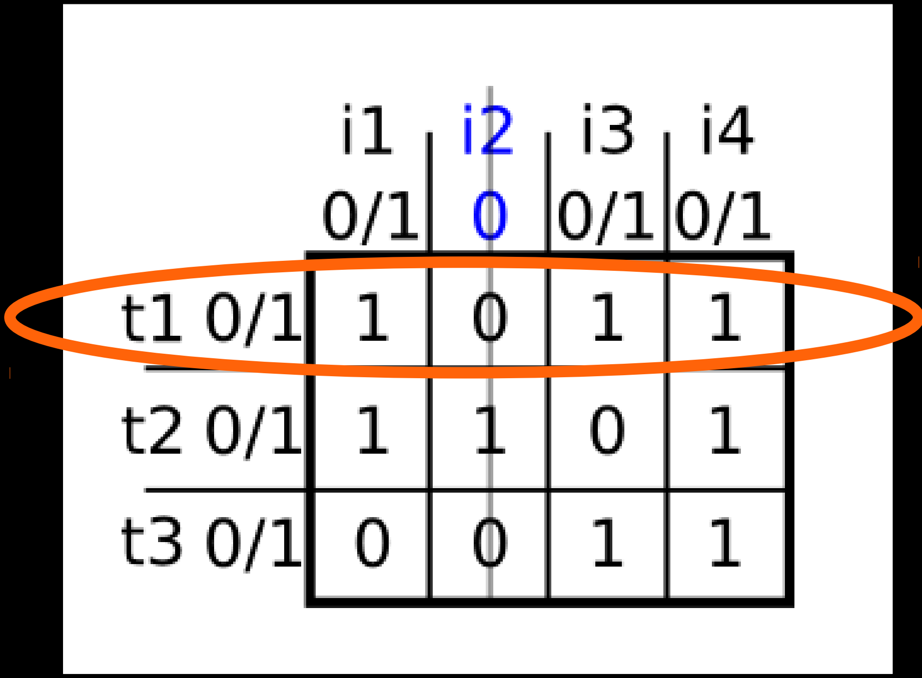
	i1 0/1	i2 0/1	i3 0/1	i4 0/1
t1 0/1	1	0	1	1
t2 0/1	1	1	0	1
t3 0/1	0	0	1	1

Search

freq >= 2: $I_i = 1 \Rightarrow \sum_t D_{ti} T_t \geq minsup$

coverage: $T_t = 1 \Leftrightarrow \sum_i I_i (1 - D_{ti}) = 0$

- propagate i2 (freq)
- propagate t1 (coverage)
  Intuition: unavoidable t1 will always be covered



	i1 0/1	i2 0	i3 0/1	i4 0/1
t1 0/1	1	0	1	1
t2 0/1	1	1	0	1
t3 0/1	0	0	1	1

Search

freq >= 2: $I_i = 1 \Rightarrow \sum_t D_{ti} T_t \geq minsup$

coverage: $T_t = 1 \Leftrightarrow \sum_i I_i (1 - D_{ti}) = 0$

- propagate i2 (freq)
- propagate t1 (coverage)

		i1	i2	i3	i4
		0/1	0	0/1	0/1
t1	1	1	0	1	1
t2	0/1	1	1	0	1
t3	0/1	0	0	1	1

Search

freq >= 2: $I_i = 1 \Rightarrow \sum_t D_{ti} T_t \geq minsup$

coverage: $T_t = 1 \Leftrightarrow \sum_i I_i (1 - D_{ti}) = 0$

- propagate i2 (freq)
- propagate t1 (coverage)
- branch i1=1
- propagate t3 (coverage)
  Intuition: t4 is missing an item of the itemset

		i1	i2	i3	i4
		1	0	0/1	0/1
t1	1	1	0	1	1
t2	0/1	1	1	0	1
t3	0/1	0	0	1	1

Search

freq >= 2: $I_i = 1 \Rightarrow \sum_t D_{ti} T_t \geq minsup$

coverage: $T_t = 1 \Leftrightarrow \sum_i I_i (1 - D_{ti}) = 0$

- propagate i2 (freq)
- propagate t1 (coverage)
- branch i1=1
- propagate t3 (coverage)
- propagate i3 (freq)
  Intuition: infrequent

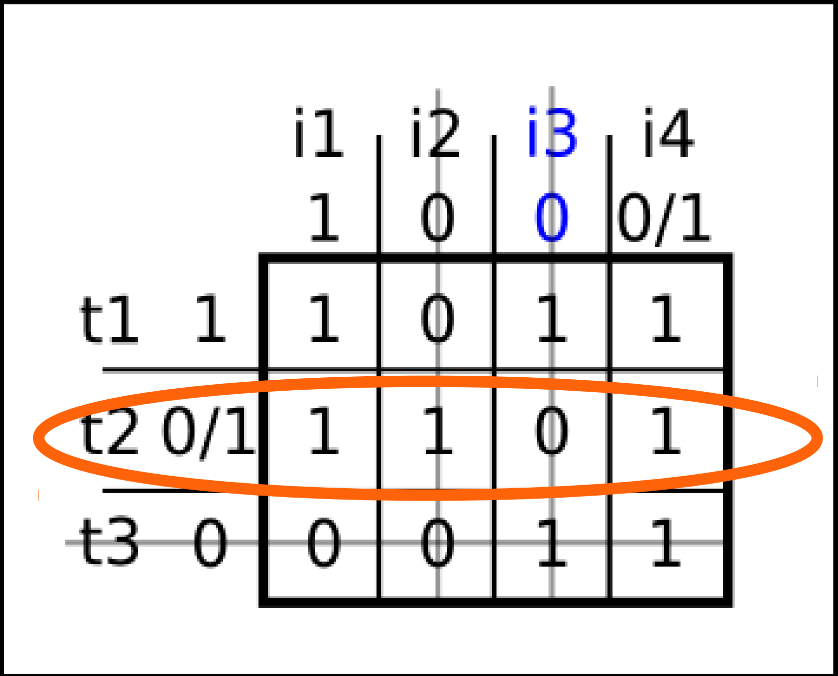
		i1	i2	i3	i4
		1	0	0/1	0/1
t1	1	1	0	1	1
t2	0/1	1	1	0	1
t3	0	0	0	1	1

Search

freq >= 2: $I_i = 1 \Rightarrow \sum_t D_{ti} T_t \geq minsup$

coverage: $T_t = 1 \Leftrightarrow \sum_i I_i (1 - D_{ti}) = 0$

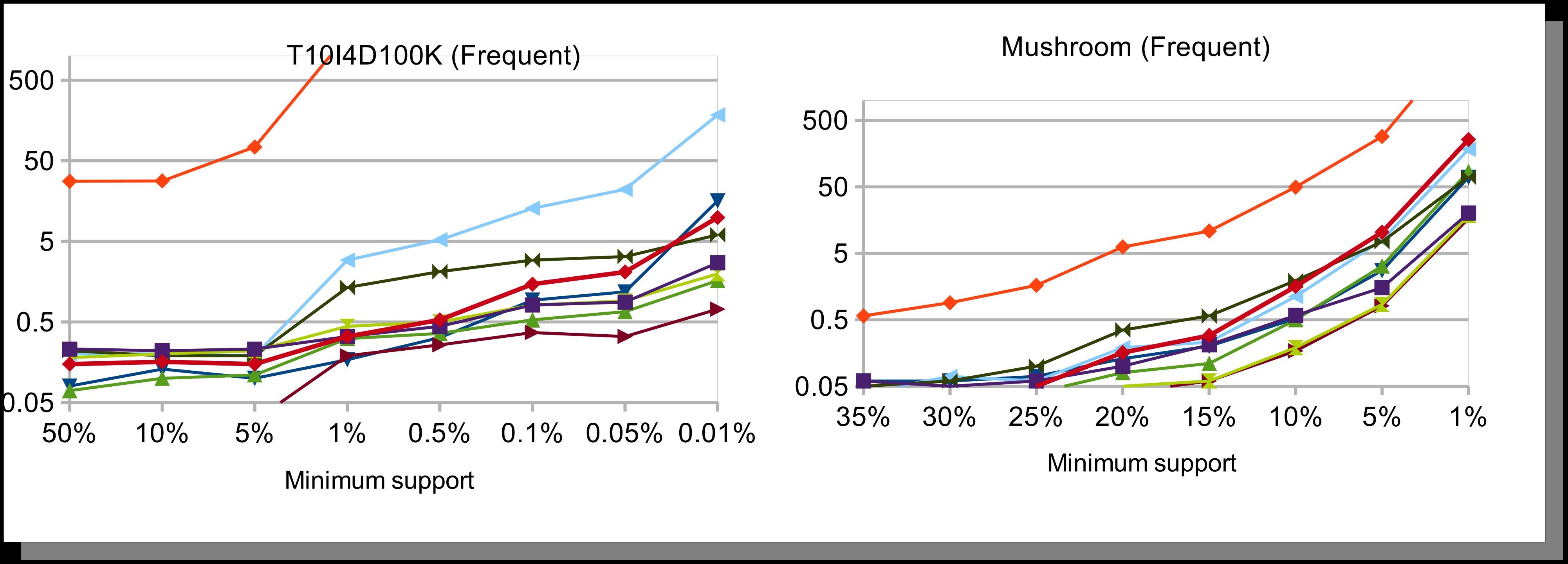
- propagate i2 (freq)
- propagate t1 (coverage)
- branch i1=1
- propagate t3 (coverage)
- propagate i3 (freq)
- propagate t2 (coverage)



Search is similar to depth-first itemset mining algorithms!

Experimental Comparison

Runtime (s)



Pattern Explosion

Constraint-based Pattern Mining

- **Given**
  - A database D with sets of items
  - A constraint $\varphi(I,D)$
- **Find**
  - **ALL** subsets of items I for which $\varphi(I,D)$ is true

Inductive Databases

- Inspired by database technology
- Use special purpose logics and solvers to find patterns under constraints

[Imielinski & Mannila, 1996]

Constraint-based Pattern Mining

- Types of constraints
  - Condensed representations
  - Supervised
  - Syntactical constraints
  - ...

Constraints: Condensed Representations

The full set of patterns can be determined from a subset

Constraints: Closed Itemsets
(Formal Concepts)

closure () = {  }

closed(I,D) ⇔ closure(I,D)=I
(Maximal rectangles)

[Pasquier et al., 1999]

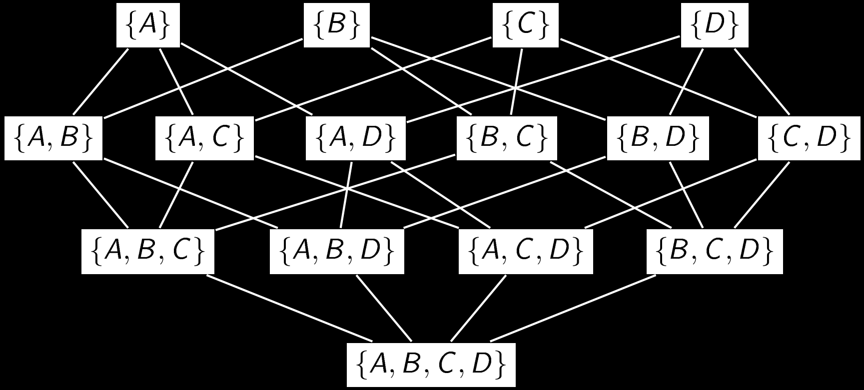
Constraints: Maximal Itemsets

(Borders in Version Spaces)

[Bayardo, 1998]

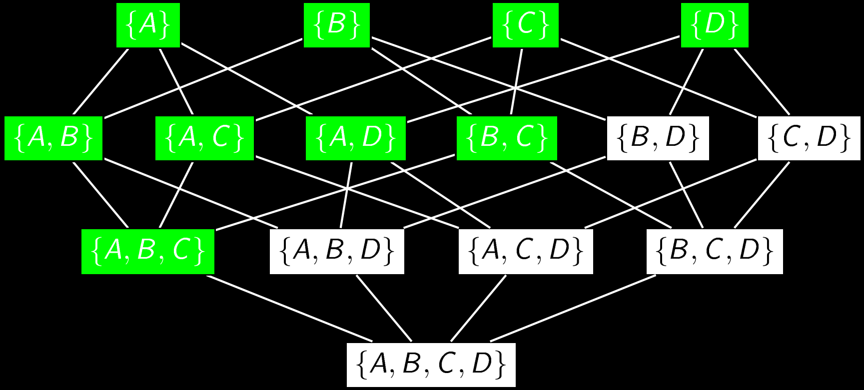
Constraints: Maximal Itemsets

(Borders in Version Spaces)



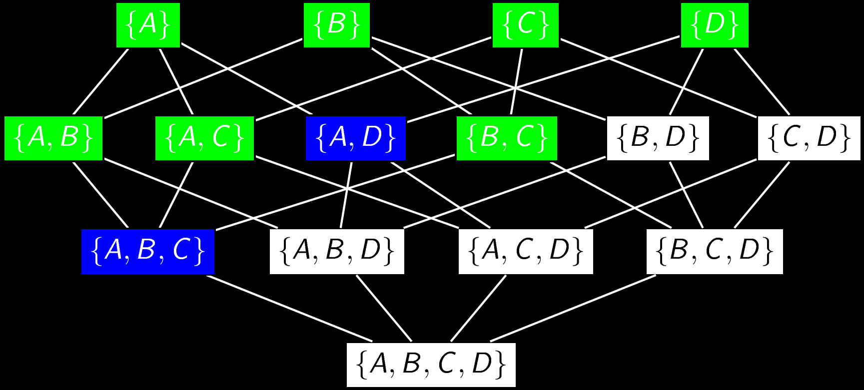
Constraints: Maximal Itemsets

(Borders in Version Spaces)



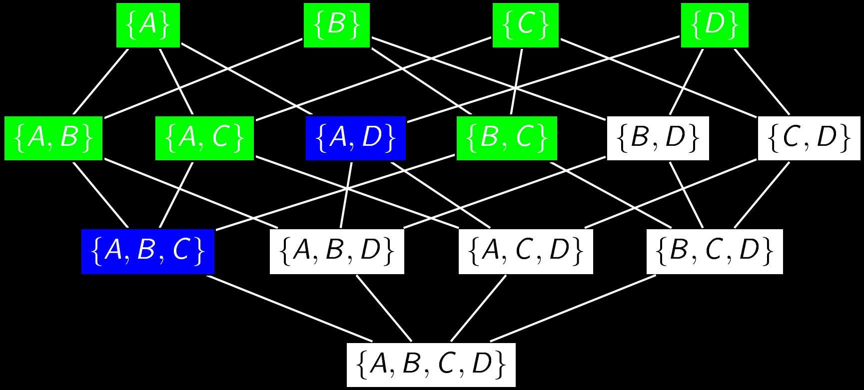
Constraints: Maximal Itemsets

(Borders in Version Spaces)



Constraints: Maximal Itemsets

(Borders in Version Spaces)



Constraints: Condensed Representations

- Maximal frequent itemset I:
  there is no $I' \supset I$ and I' frequent
- Closed itemset I:
  there is no $I' \supset I$ and support(I')=support(I)
- Free itemset I:
  there is no $I' \subset I$ and support(I')=support(I)

Search

- Many specialized algorithms developed in data mining (breadth-first, depth-first, ...)
- Can CP be a general framework?

Condensed Representations in CP

- Frequent Itemset Mining

$$I_i = 1 \Rightarrow \sum_t D_{ti} T_t \geq minsup$$
$$T_t = 1 \Leftrightarrow \sum_i I_i (1 - D_{ti}) = 0$$

- Maximal Frequent Itemset Mining

$$I_i = 1 \Leftrightarrow \sum_t D_{ti} T_t \geq minsup$$
$$T_t = 1 \Leftrightarrow \sum_i I_i (1 - D_{ti}) = 0$$

- Closed Itemset Mining

$$I_i = 1 \Rightarrow \sum_t D_{ti} T_t \geq minsup$$
$$I_i = 1 \Leftrightarrow \sum_t T_t (1 - D_{ti}) = 0$$

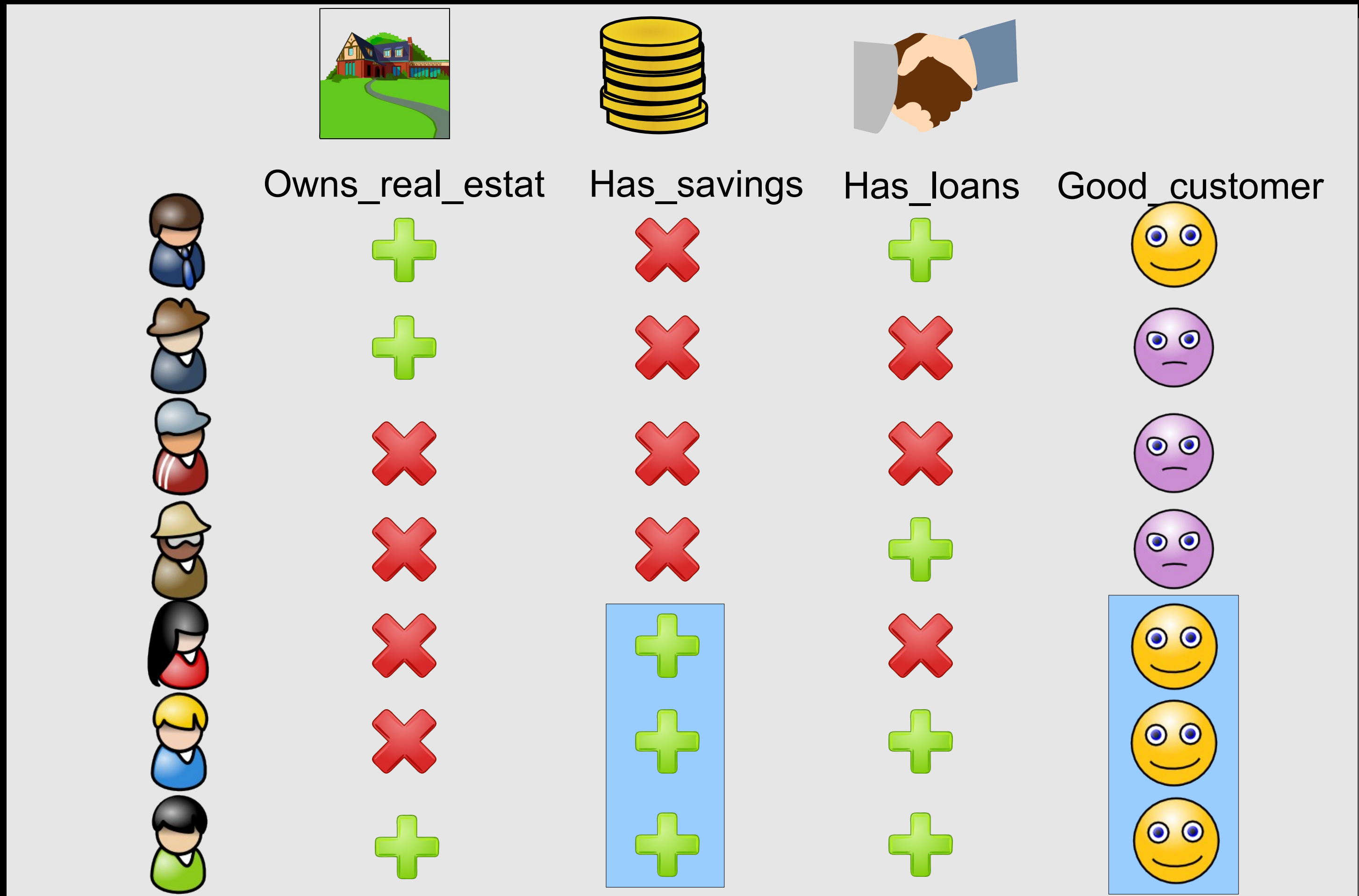
- (δ-)Closed Itemset Mining

$$T_t = 1 \Leftrightarrow \sum_i I_i (1 - D_{ti}) = 0$$
$$I_i = 1 \Rightarrow \sum_t D_{ti} T_t \geq minsup$$
$$I_i = 1 \Leftrightarrow \sum_t T_t (1 - \delta - D_{ti}) \leq 0$$

Emulates...

- Eclat
- Mafia
- LCM

Itemsets in Supervised Data

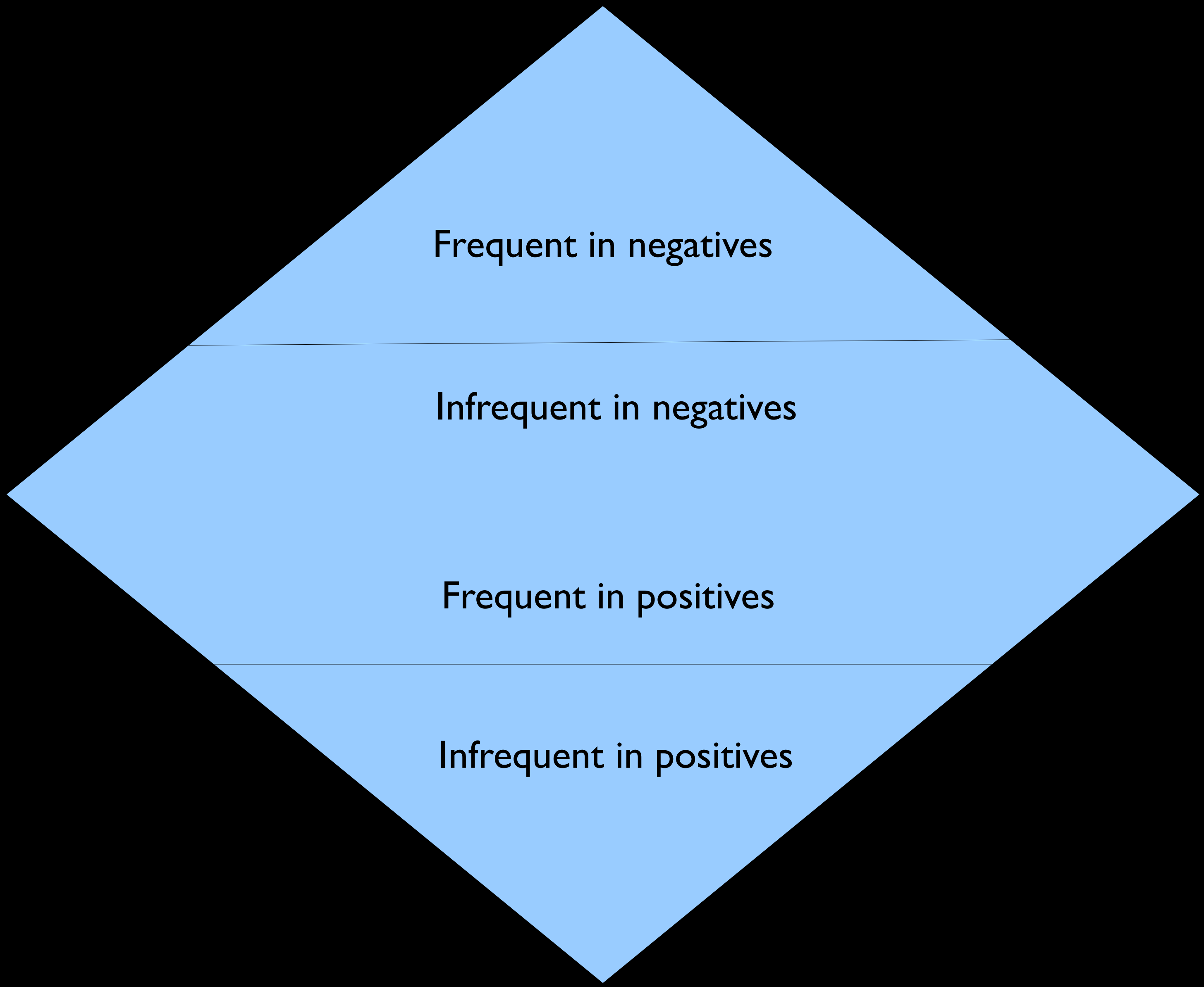


Contingency Table

TP: 3 (=p)	FP: 0 (=n)	3
FN: 1	TN: 3	4
P: 4	N: 3	

[Nijssen et al., 2009]

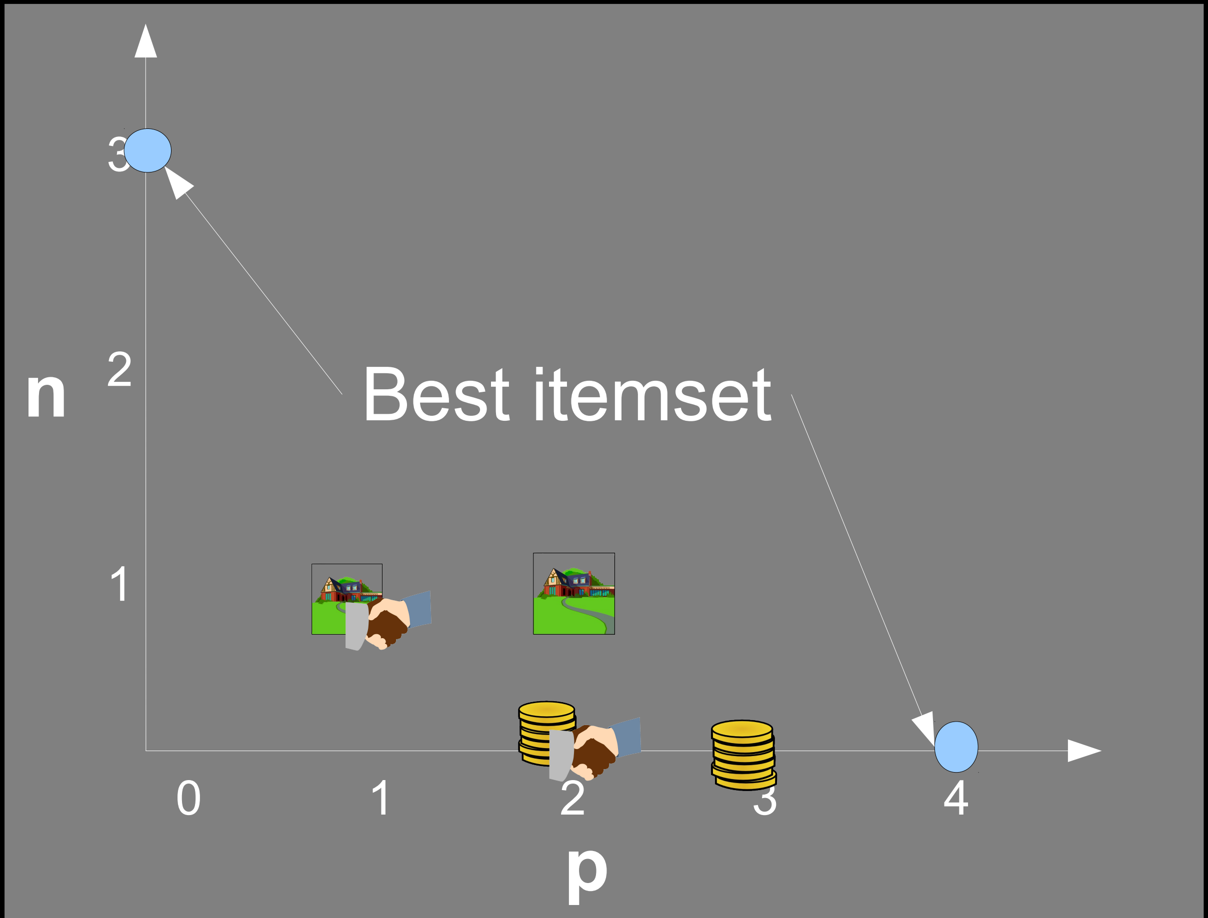
Itemsets in Supervised Data



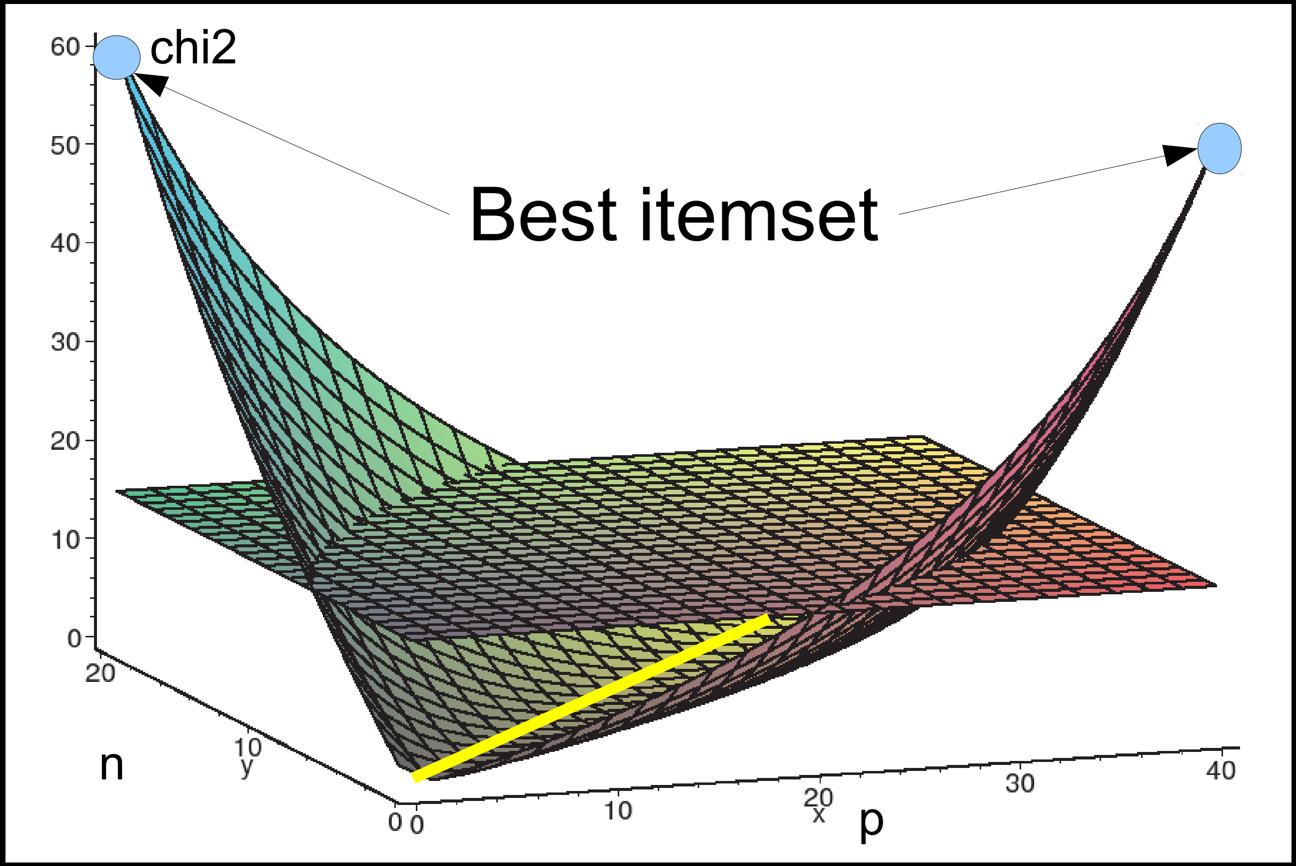
Itemsets in Supervised Data

Contingency Table

TP: 3 (=p)	FP: 0 (=n)	3
FN: 1	TN: 3	4
P: 4	N: 3	



Itemsets in Supervised Data



Many correlation functions (chi2, fisher, inf. gain)

are convex and zero on the diagonal

Itemsets in Supervised Data

- Again, many different algorithms
- In CP:

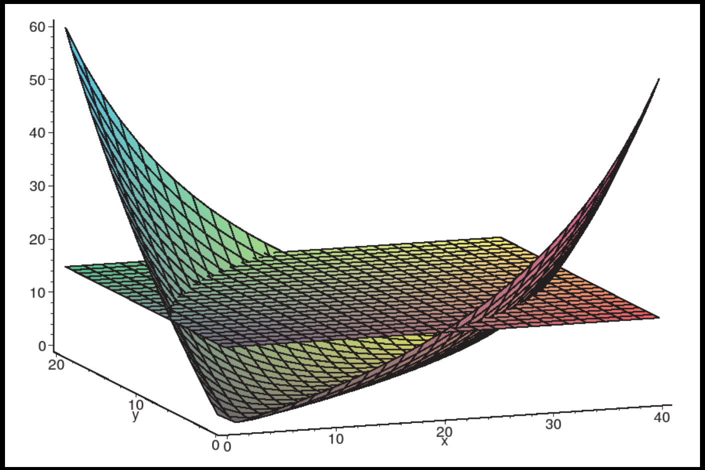
$$I_i = 1 \Rightarrow f\left(\sum_{t \in T^+} D_{ti} T_t, \sum_{t \in T^-} D_{ti} T_t\right) \geq mincorr$$

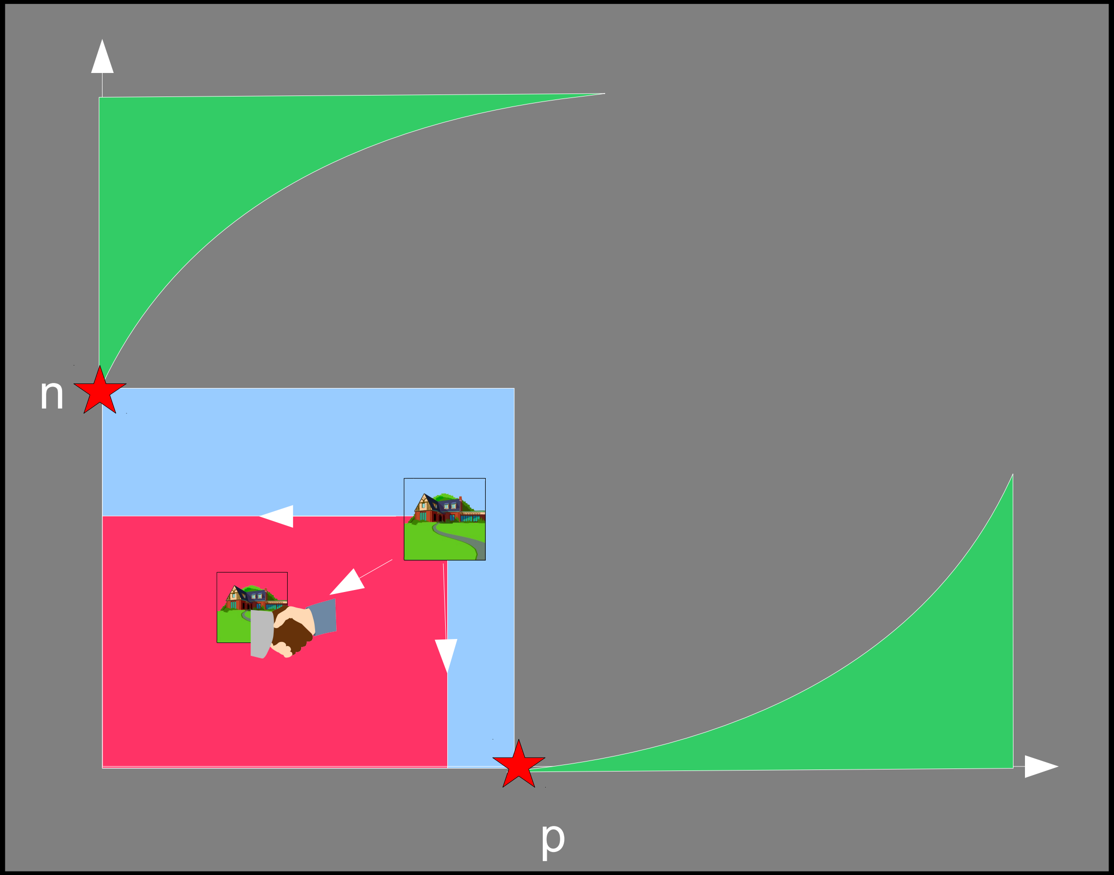
$$T_t = 1 \Leftrightarrow \sum_i I_i (1 - D_{ti}) = 0$$

Itemsets in Supervised Data

General to specific search

- Adding an item will give equal or lower *p* and *n*

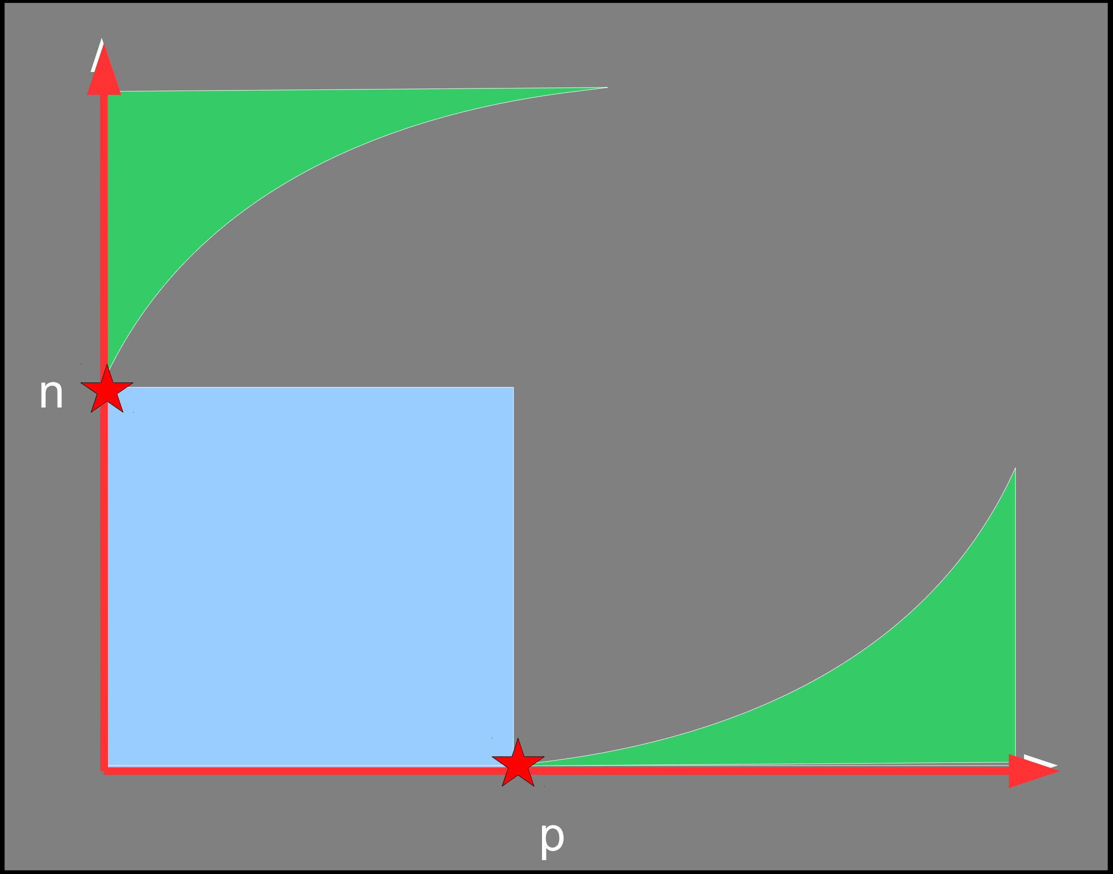




Itemsets in Supervised Data

Key observation: unavoidable transactions

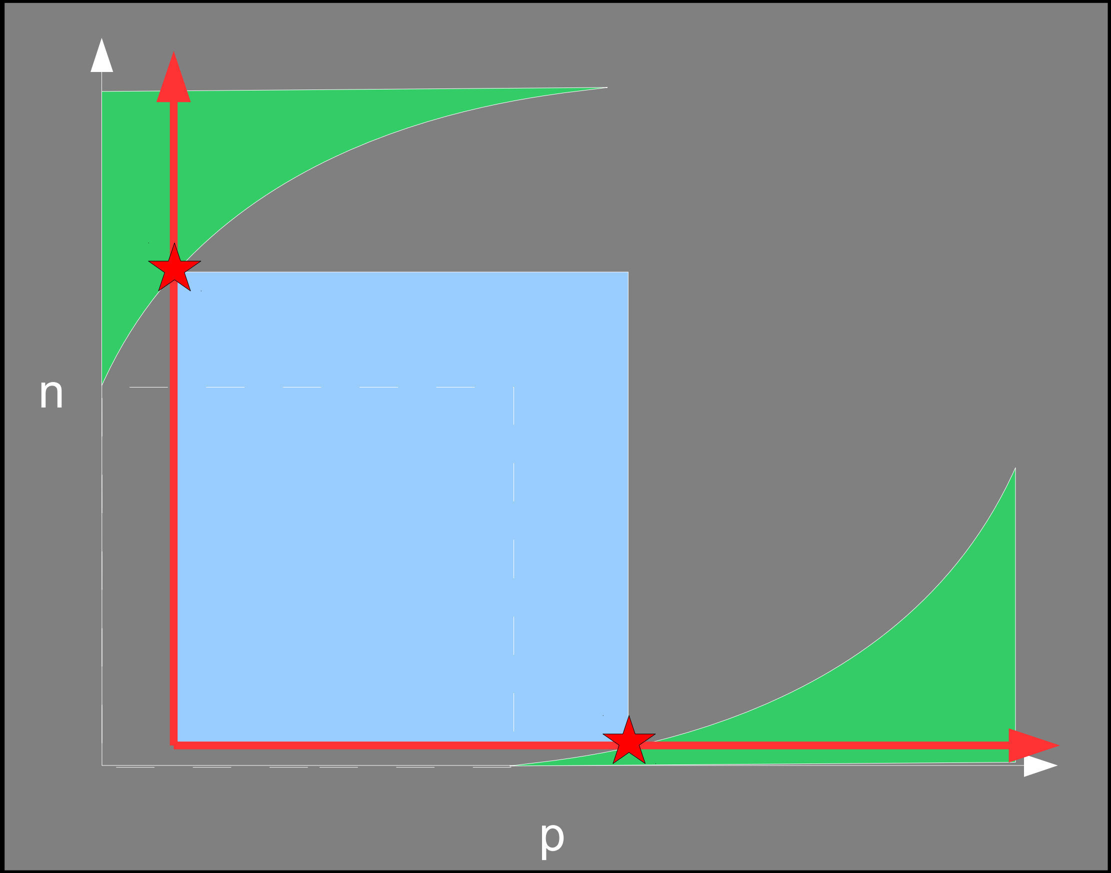
	i1 0/1	i2 0	i3 0/1	i4 0/1
t1 0/1	1	0	1	1
t2 0/1	1	1	0	1
t3 0/1	0	0	1	1



Itemsets in Supervised Data

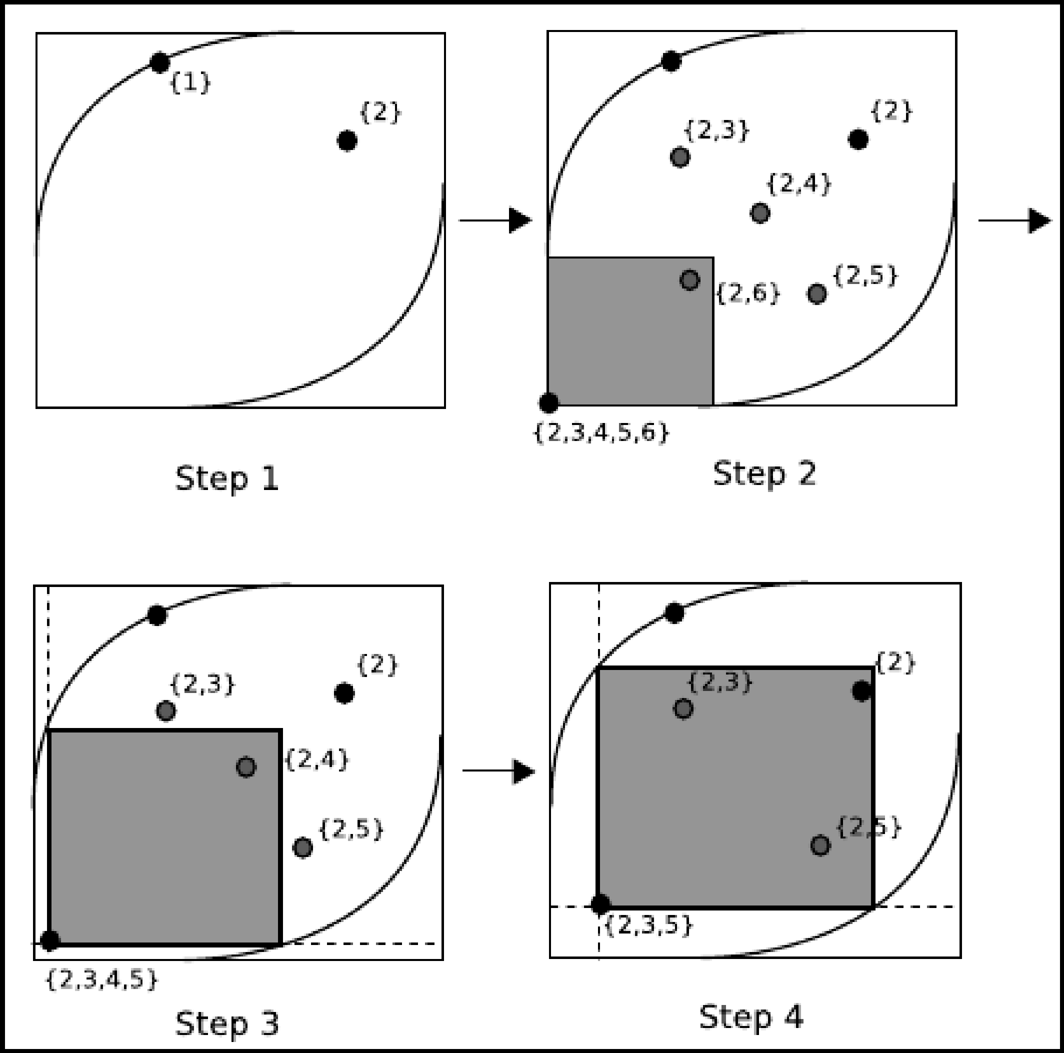
Key observation: unavoidable transactions

	i1 0/1	i2 0	i3 0/1	i4 0/1
t1 0/1	1	0	1	1
t2 0/1	1	1	0	1
t3 0/1	0	0	1	1



Itemsets in Supervised Data

iterative propagation:



Experimental Comparison

Name	corrmine	cimcp	ddpmine	lcm
anneal	0.02	0.22	22.46	7.92
australian-credit	0.01	0.30	3.40	1.22
breast-wisconsin	0.03	0.28	96.75	27.49
diabetes	0.36	2.45	–	697.12
german-credit	0.07	2.39	–	30.84
heart-cleveland	0.03	0.19	9.49	2.87
hypothyroid	0.02	0.71	–	>
ionosphere	0.24	1.44	–	>
kr-vs-kp	0.02	0.92	125.60	25.62
letter	0.65	52.66	–	>
mushroom	0.03	14.11	0.09	0.03
pendigits	0.18	3.68	–	>
primary-tumor	0.01	0.03	0.26	0.08
segment	0.06	1.45	–	>
soybean	0.01	0.05	0.05	0.02
splice-1	0.05	30.41	1.86	0.02
vehicle	0.07	0.85	–	>
yeast	0.80	5.67	–	185.28
avg. when found:	*0.15*	*6.55*	*28.88+*	*81.54+*

CP for Pattern Mining

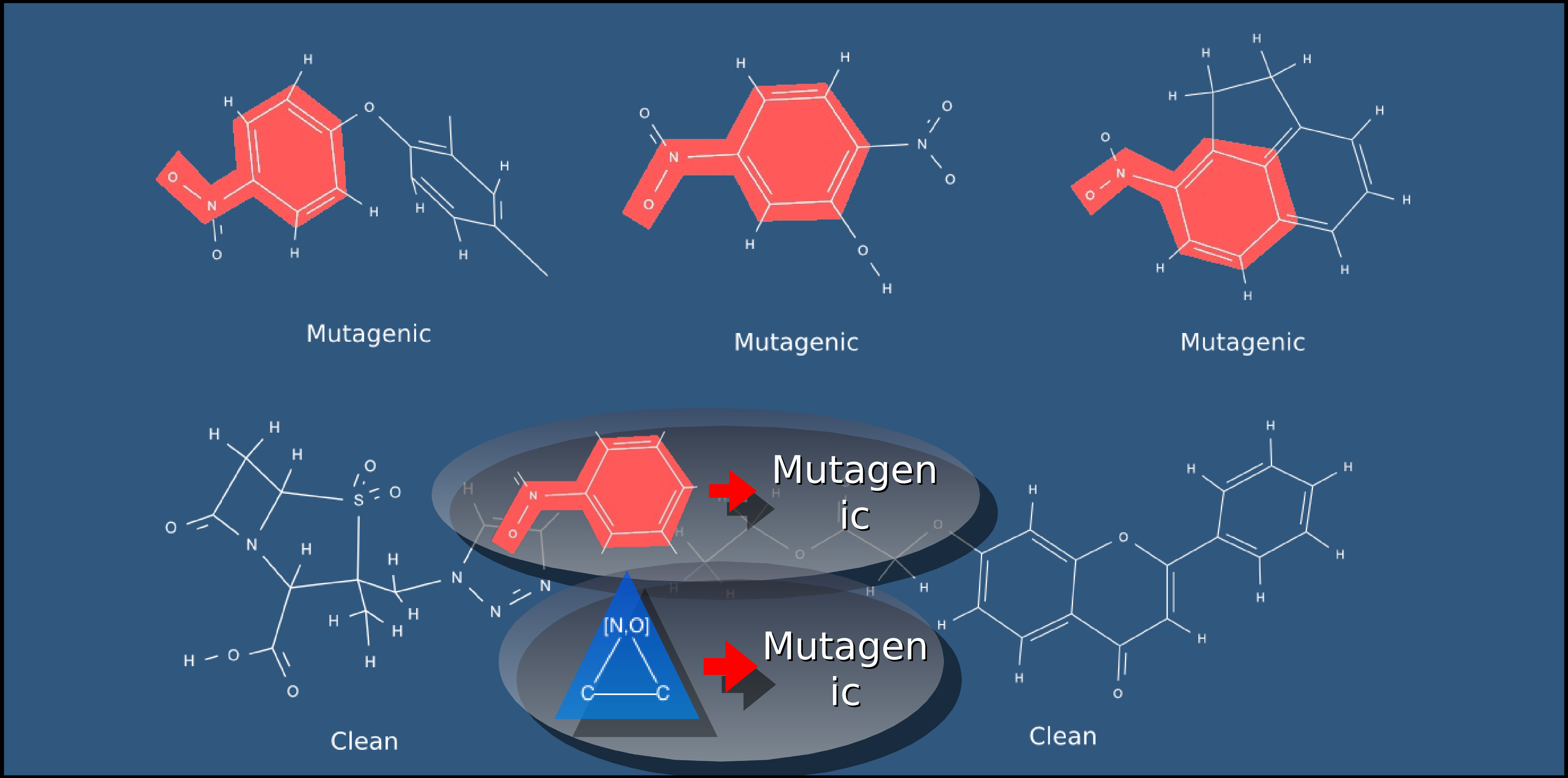
- Promising results
  - More general framework: combining constraints, formalizing new constraints
  - Sometimes more efficient

Challenges

- Other pattern languages
- Pattern *set* mining

Other Pattern Languages

Graphs



[Inokuchi & Washio, 2003]

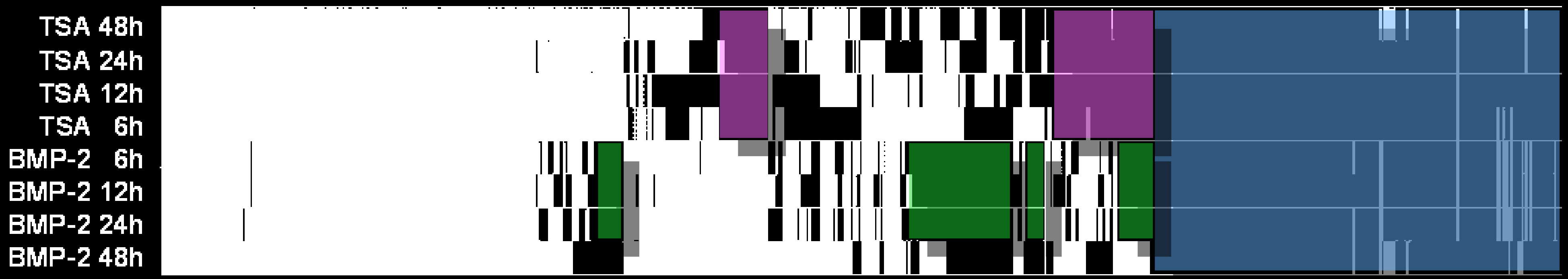
Other Pattern Languages

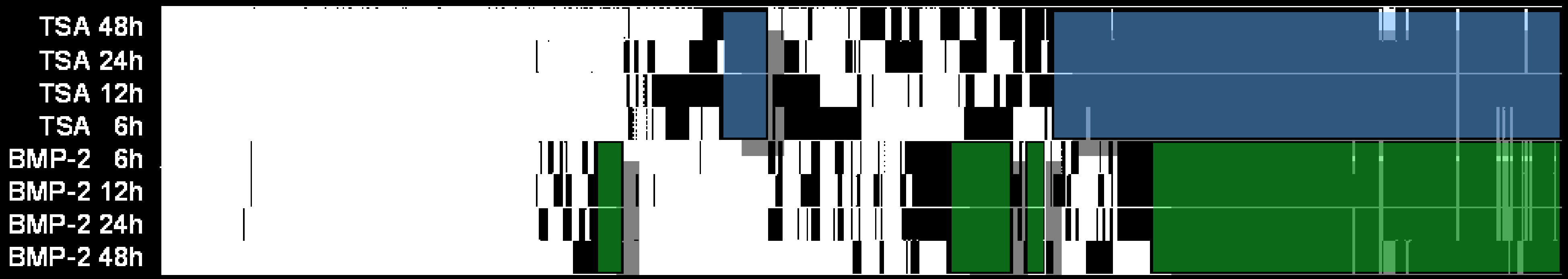
- Graphs *[Inokuchi & Washio, 2003]*
- Trees *[Zaki, 2002]*
- Strings *[Fischer & Kramer, 2006]*
- Sequences *[Agrawal & Srikant, 1995]*
- Clausal formulas *[Dehaspe & De Raedt, 1997]*
- ...

See also http://usefulpatterns.org/msop/

Pattern Set Mining

- Constraints on individual patterns do not solve the pattern explosion





Aim: to find a small *set* of patterns that *together* are representative / useful

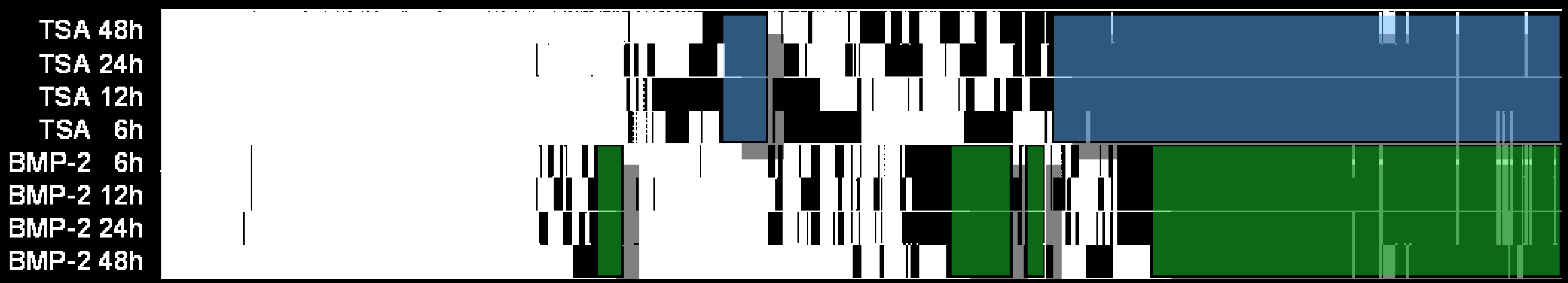
Pattern Set Mining

- **Given**
  - A database D with sets of items
  - A constraint $\varphi(I,D)$ on patterns *I*
  - A constraint $\Phi(\mathbf{I},D)$ on a **set** of patterns **I**
  - An optimization criterion $f(\mathbf{I},D)$ on a **set** of patterns **I**
- **Find** the set of patterns **I** such that
  - $f(\mathbf{I},D)$ is maximized
  - Each *I* in **I** satisfies $\varphi(I,D)$
  - **I** satisfies $\Phi(\mathbf{I},D)$

[De Raedt & Zimmermann, 2007]

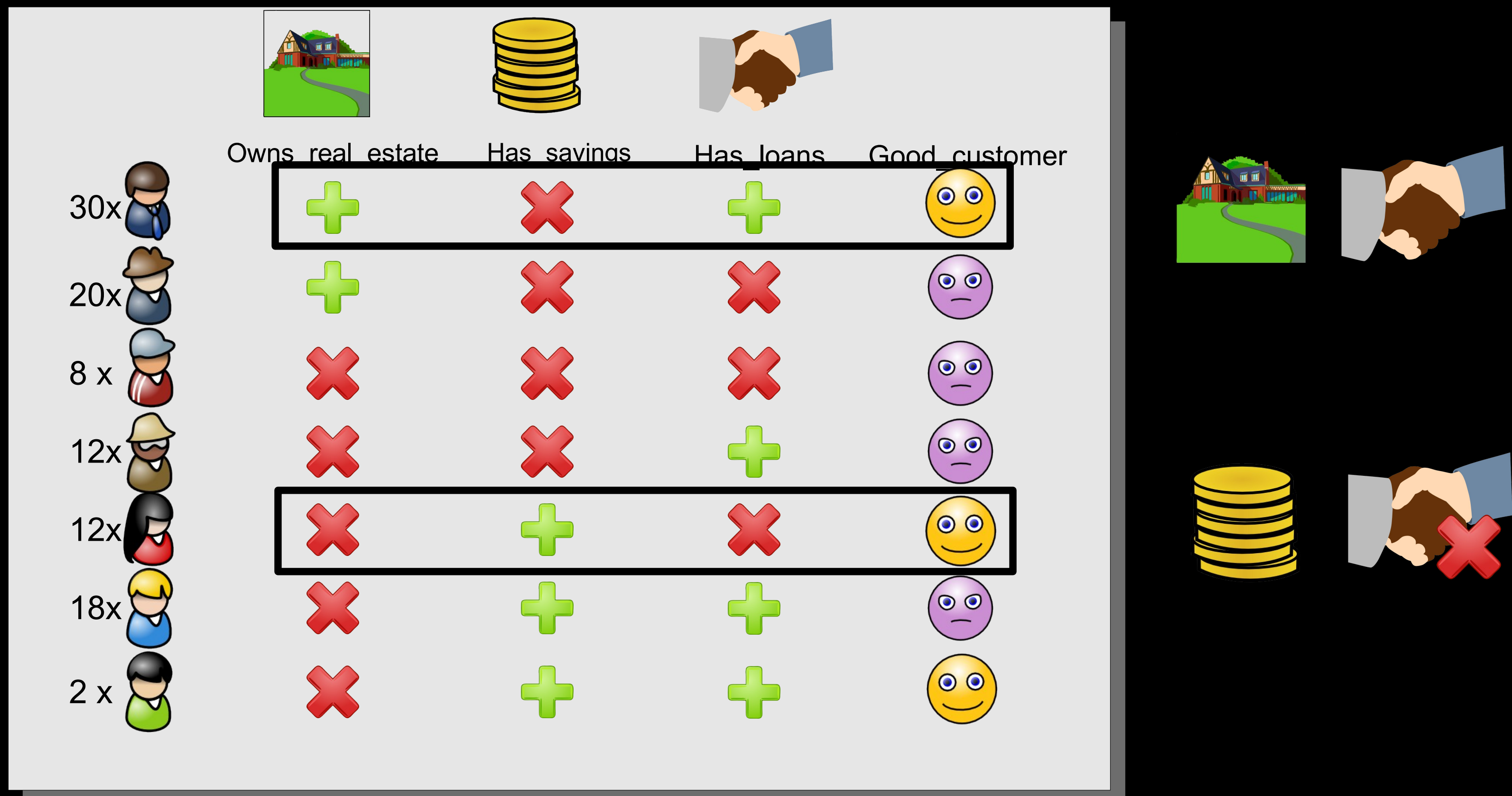
Pattern Set Mining

- Co-clustering (aka tiling): "covering the black parts of a matrix with rectangles"
  → Many different formalizations (overlap/size/tolerance for errors/...)



Pattern Set Mining

- Rule-based classification: “predict examples”
  → Many different formalizations (error/ordering of patterns/label in rules/...)



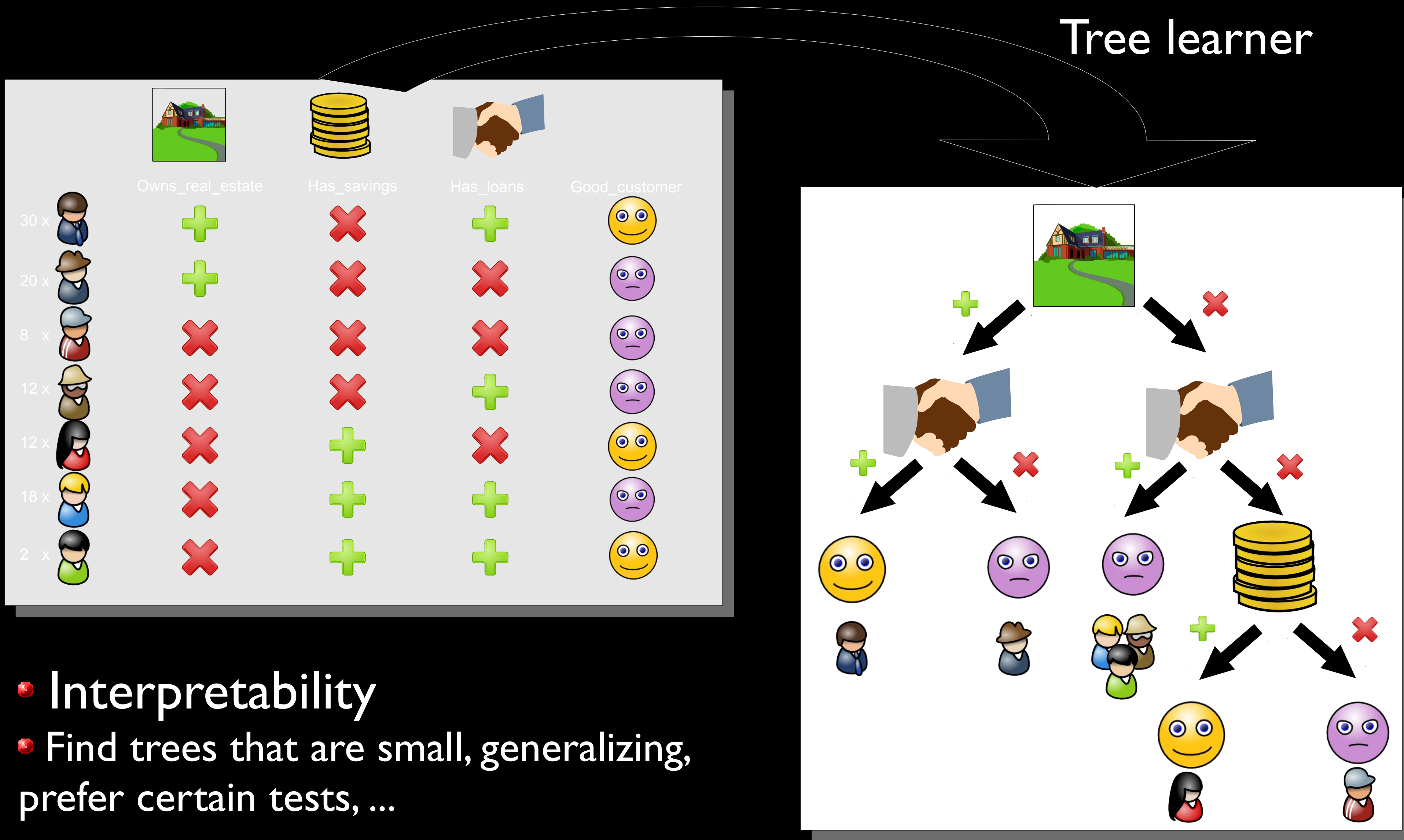
Pattern Set Mining

- A general declarative approach?
  - CP systems (Gecode) on declarative formalization of problems with **fixed** pattern set size *[Guns et al.]* (does not scale)
  - SAT solvers (Minisat) on declarative formalization of problems with **fixed** pattern set size *[Cremilleux et al.]* (does not scale)
  - Local search systems (Comet) (scales better, but still cumbersome when pattern set size is not fixed in advance) *[Guns et al.]*

Decision Trees

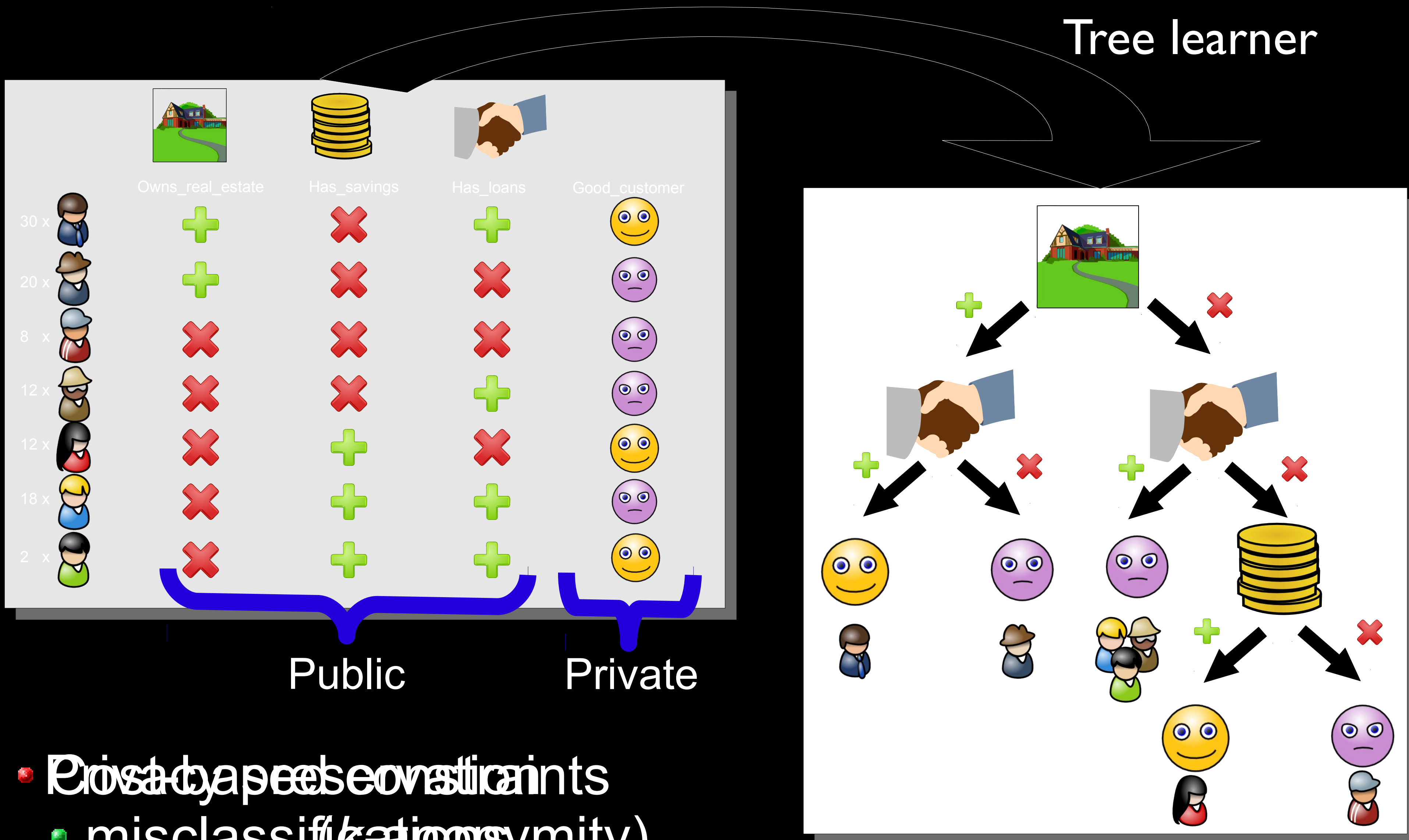
- Special type of classifier for which more general solvers have been developed
- **Most common approach:**
  use heuristics to build a tree
  → no constraints
  → no global optimization criterion
- In some cases unsatisfactory

What is a Decision Tree?



- Interpretability
- Find trees that are small, generalizing, prefer certain tests, ...

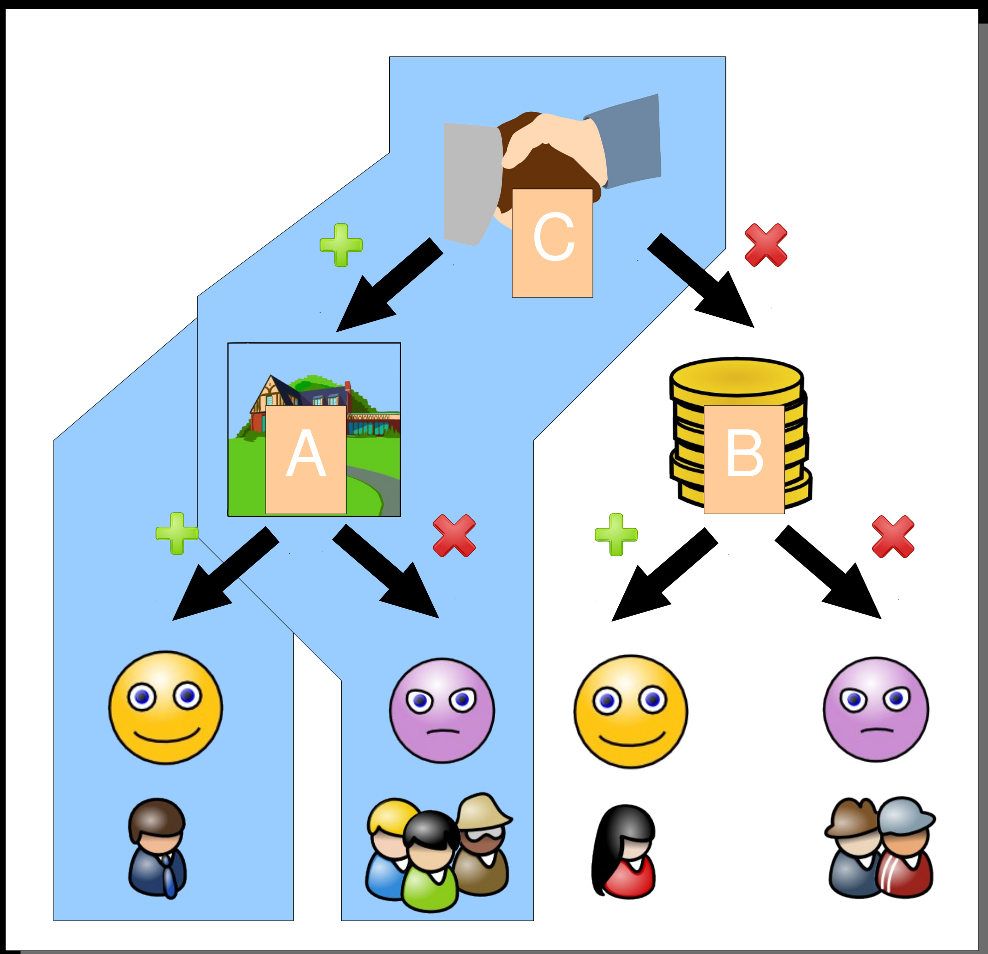
What is a Decision Tree?



- Privacy constraints
  - (k-anonymity)
- Cost-based construction
  - misclassification

Finding Decision Trees: DL8

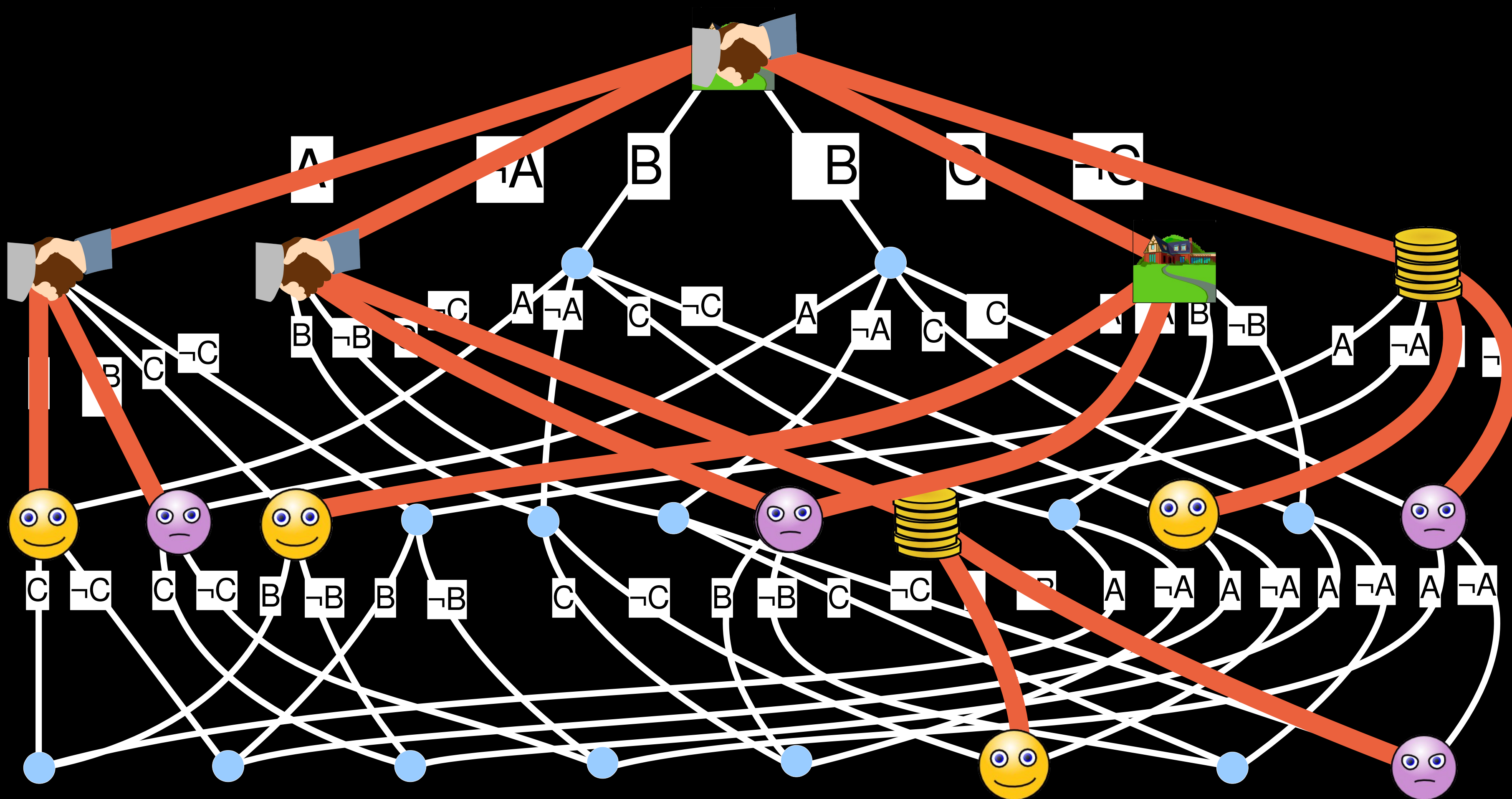
- Support constraints on leafs → exploit relationship to itemset mining



Itemset	Tids
{ C, A }	
{ C, ¬A }	
{ ¬C, B }	
{ ¬C, ¬B }	

[Nijssen & Fromont, 2007, 2010]

Finding Decision Trees: DL8



- Decision Trees are hidden in the lattice; if lattices is stored, one can do *dynamic programming*

Finding Decision Trees: Any Time Algorithm

- Discover the smallest 100% accurate decision tree
- First proposed solution:
  - Greedy algorithm
  - Sample from space of trees to determine expected size after a split (increased sample size → better estimate)
    - Sampling biased by traditional heuristics
- Second proposed solution:
  - Use the first proposed solution to iterative improve subtrees of a tree by using more resources (sample size)

[Esmeir & Markovitch, 2007]

Finding Decision Trees: SAT solvers, CP systems, LP

- Discover the smallest 100% accurate decision tree by means of encoding
- SAT encoding: $O(kn^2m^2 + nk^2 + kn^3)$ space.
  (n = maximum number of nodes in *complete* tree, k = number of features, m = number of examples)
- CP encoding:
  - Variables for tree nodes
  - Variables for examples in tree nodes
  - Constraints enforcing tree structure (global constraint), binary splits, examples in tree nodes, leafs are pure (logical constraints)
  - Search heuristics, **random restarts**
  - Improvement by means of LP with m^2 variables (on small sets)

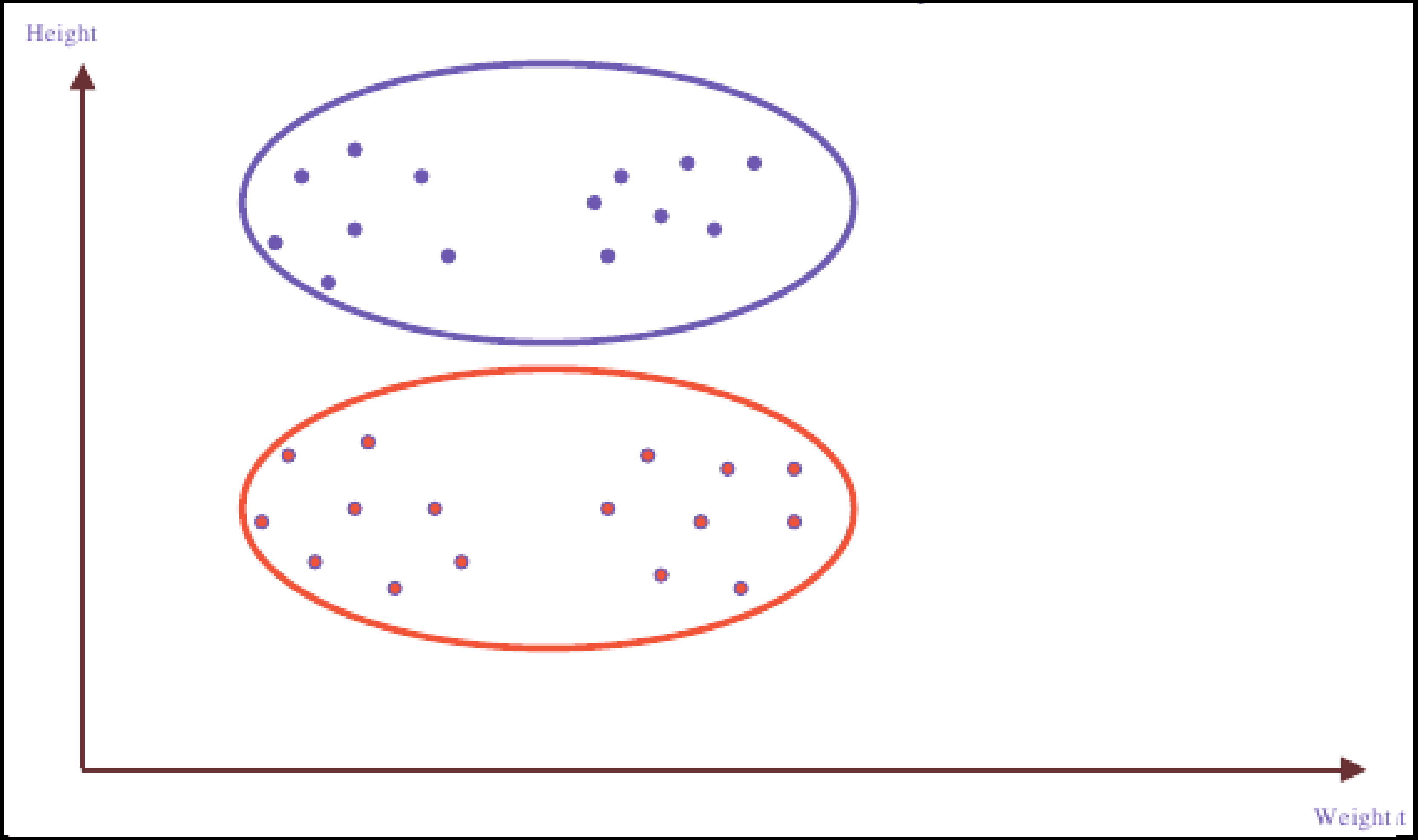
[Bessiere, Hebrard, O'Sullivan, 2009]

Clustering

- What is clustering?
- What are constraints in clustering?
- Using solvers for clustering

Clustering

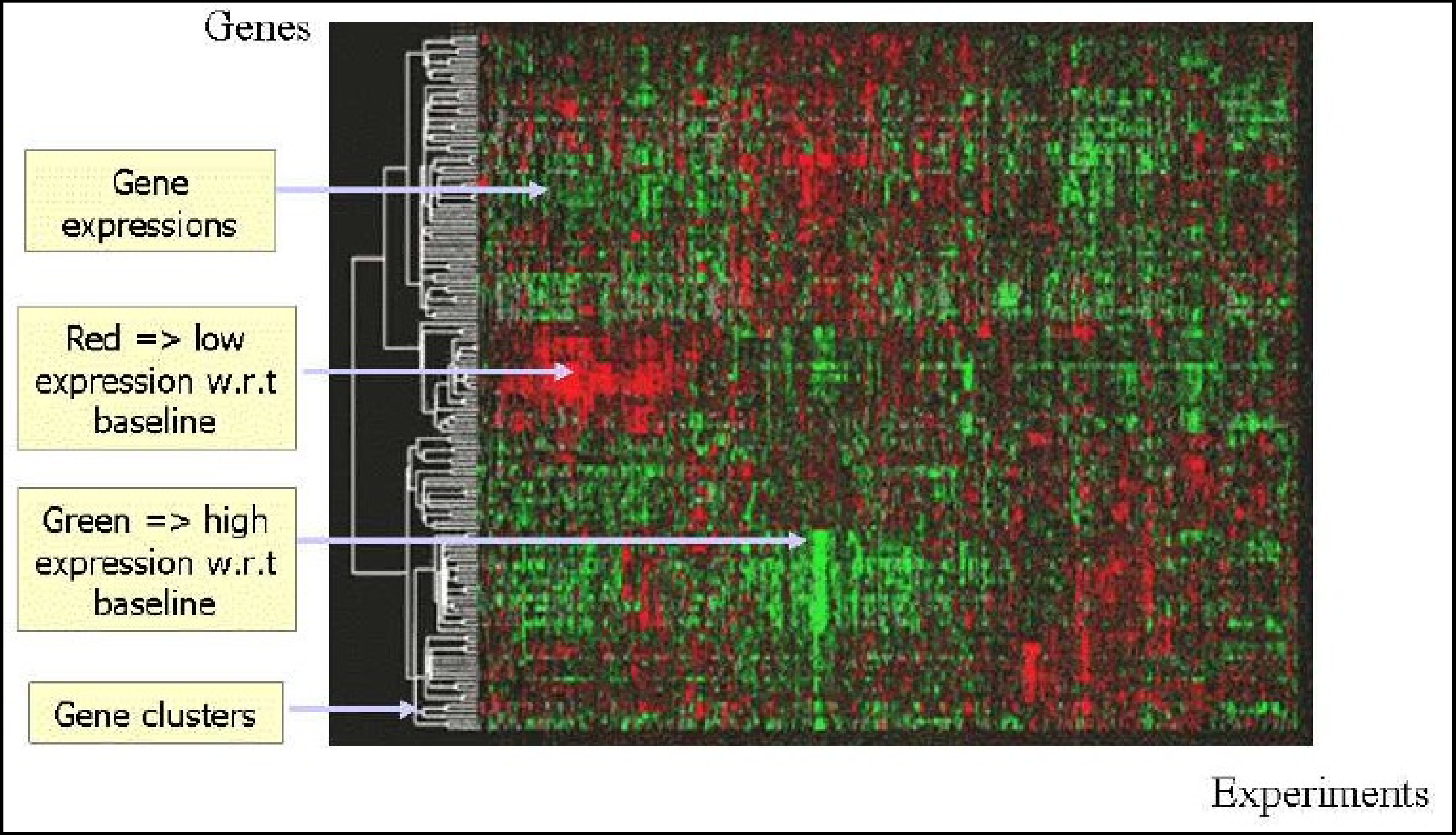
- Fixed number of clusters



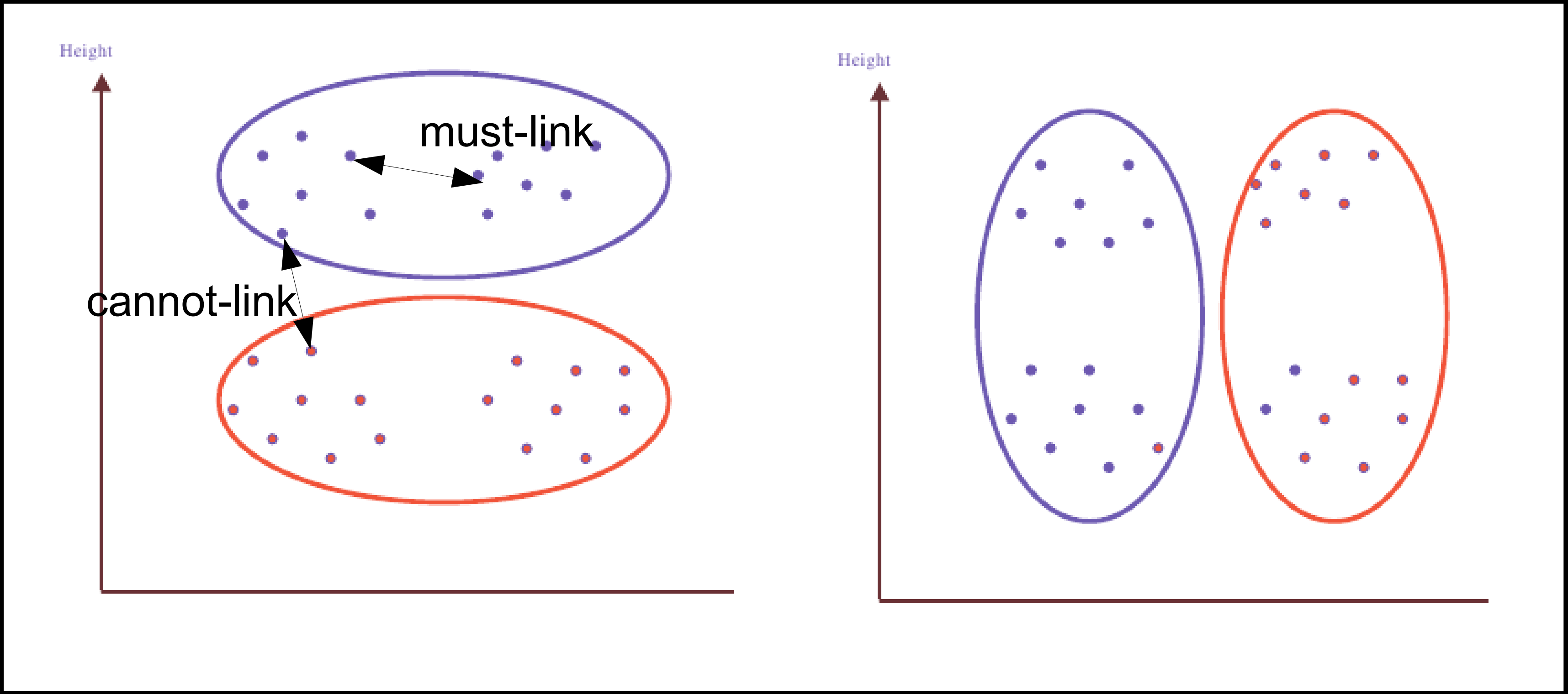
[Basu & Davidson 2006, 2011]

Clustering

- Hierarchical clustering



Constraints in Clustering



Express preferences directly
Help clustering algorithm finding the “right” solution
Find alternative clusterings (subspace clustering)
Semi-supervised learning

Constraints in Clustering

- In hierarchical clustering:
  - Must-link-before constraint
    a and b must both be in the same cluster before being merged with c
  - Level specific constraints
    a and b can only be merged in the top n layers

Algorithms

- Traditional algorithm + modified distance function
  either learned, or hand-tuned

- Traditional algorithm + tweaks to enforce hard constraints (i.e. must-link constraints)

- New algorithms
  few

Hierarchical Clustering

- Traditional algorithm without constraints: *iteratively merge the two clusters that are most near*
- Modified algorithm:
  1. *Encode constraints in Horn clauses*
  2. *Calculate valid merges, i.e. merges that can lead to a valid solution*
  3. *Select most promising merge*
  4. *Go to 2.*
- Valid merges are calculated in polynomial time $O(n^2)$

[Gilpin & Davidson, 2011]

Intermediate Conclusions

- Many problems in data mining can be seen as constraint optimisation problems
- Scalability with respect to data size (both rows and columns) is important
- Most algorithms are not generic algorithms
- There are opportunities to exploit constraint solving technology in data mining

How ML might help CP

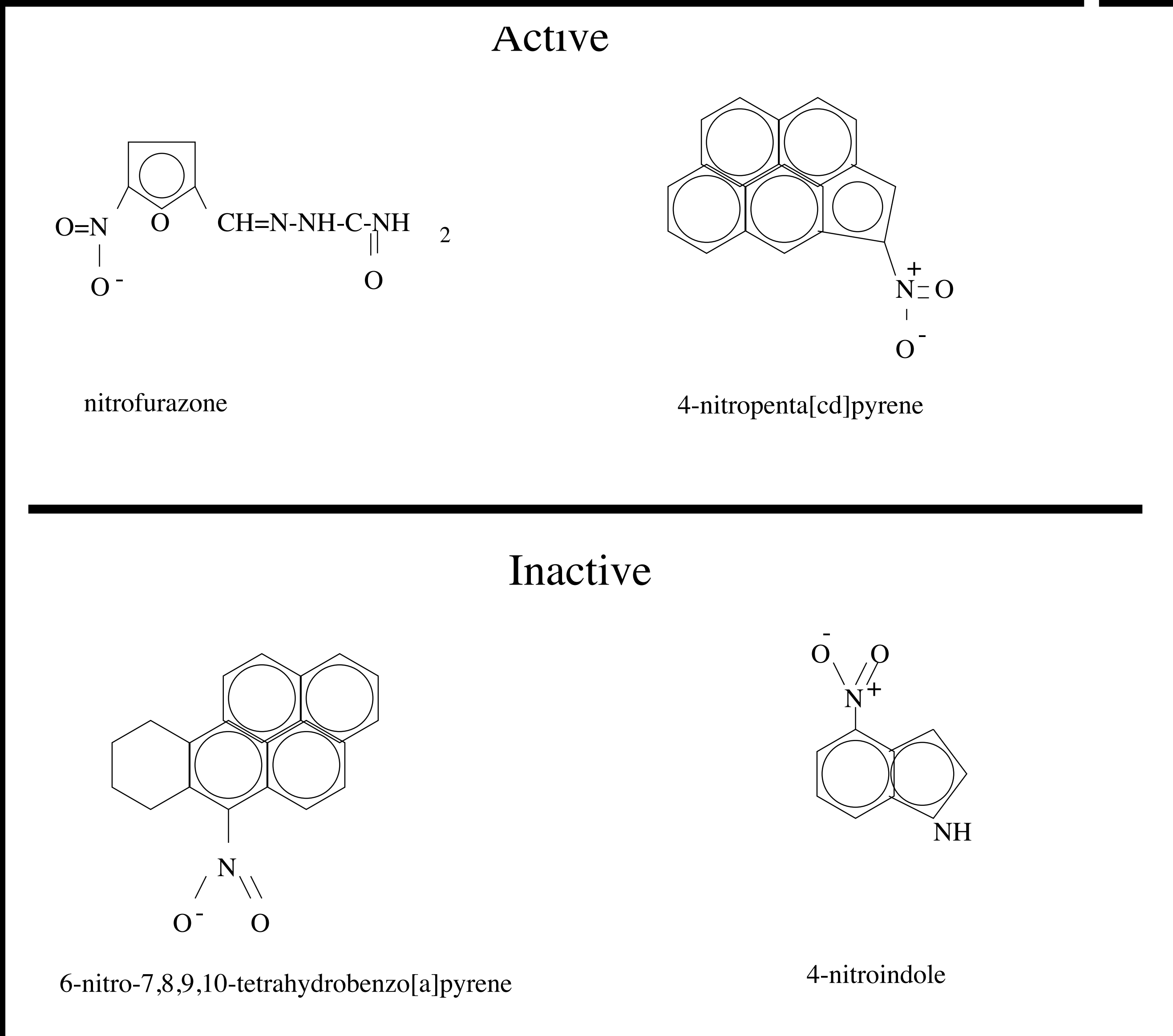
Machine Learning for CP

CSP (V,D,C,f) (f: Optimisation function)

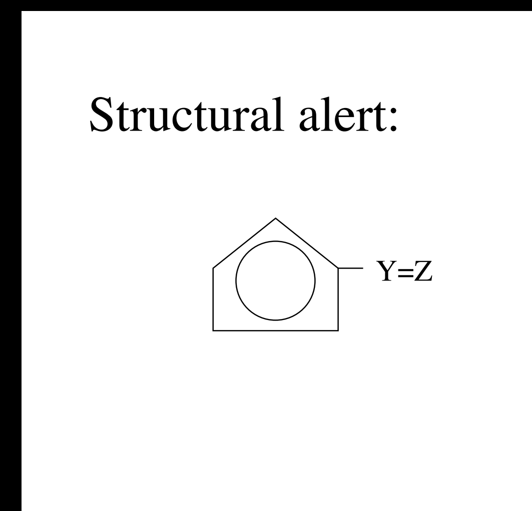
At least three interpretations

- Learning CSP(V,D,C,f) from examples
- Learning to solve for better performance
    - “clause” learning etc. (speed-up learning, explanation based learning)
    - learning portfolio’s of solvers (meta-learning, preference learning)

Structure Activity Relationship Prediction



[Srinivasan et al. AIJ 96]



Data = Set of Small Graphs

Machine Learning

Given

- a space of possible instances X
- an unknown target function $f: X \rightarrow Y$
- a hypothesis space L containing functions $X \rightarrow Y$
- a set of examples $E = \{ (x, f(x)) \mid x \in X \}$
- a loss function *loss*(h,E) $\rightarrow \mathbb{R}$

Find $h \in L$ that minimizes *loss*(h,E)

supervised

Classification

Given - Molecular Data Sets

- a space of possible instances X -- Molecular Graphs
- an unknown target function f: $X \rightarrow Y$ -- {Active,Inactive}
- a hypothesis space L containing functions $X \rightarrow Y$ -- L= {Active iff structural alert s covers instance $x \in X | s \in X$ }
- a set of examples $E = \{ (x, f(x)) \mid x \in X \}$
- a loss function *loss*(h,E) $\rightarrow \mathbb{R}$ $|\{ x \in E \mid f(x) \neq h(x)\}|$

Find $h \in L$ that minimizes *loss*(h,E)

If classes = {positive, negative} then this is concept-learning

Regression

Given - Molecular Data Sets

- a space of possible instances X -- Molecular Graphs
- an unknown target function f: X → Y -- $\mathbb{R}$
- a hypothesis space L containing functions X → Y -- a linear function of some features
- a set of examples E = { (x, f(x)) | x ∈ X }
- a loss function *loss*(h,E) → $\mathbb{R}$

$$\sqrt{\sum_{x \in E} f(x)^2 - h(x)^2}$$

- **Find** h ∈ L that minimizes *loss*(h,E)

Learning Probabilistic Models

Given

- a space of possible instances X
- an unknown target function P: X → Y   Y=[0,1]
- a hypothesis space L containing functions X → Y (graphical models)
- a set of examples E = { (x, _) | x ∈ X }   generative
- a loss function *loss*(h,E) → ℝ $\prod_{e \in E} P(e|h)$

Find h ∈ L that minimizes *loss*(h,E)   maximize likelihood

generative

Boolean Concept-Learning

$X = \{(X_1, ..., X_n) \mid X_i = 0 / 1\}$

$Y = \{+,-\}$

L = boolean formulae

loss(h,E) = training set error

$= | \{e \mid e \in E, h(e) \neq f(e)\} | \; / \; |E|$

sometimes required to be 0

Simplest setting for learning, compatible with DM part and with CP

Boolean concept-learning

	1	2	3	4	5		
ex 1	0	1	0	1	0	..	+
ex 2	1	1	1	1	1		+
ex 3	0	1	1	0	0		-
ex 4	1	0	0	1	0		-
...							

X_2 and X_4

Dimensions

Given

- a space of possible instances X
- an unknown target function f: X → Y
- a hypothesis space L containing functions X → Y

  k-CNF ?
  DNF ? etc
- a set of examples E = { (x, f(x)) | x ∈ X }

  pos and neg ?
  or pos only
- a loss function *loss*(h,E) → ℝ loss/error=0 required ?

Find h ∈ L that minimizes *loss*(h,E)

ability to ask questions ?

Why boolean concept-learning ? constraint networks

(V_1, V_2, V_3)	$V_1 < V_2$	$V_1 > V_2$	$V_1 = V_2$	$V_1 < V_3$
(1,2,3)	1	0	0	1
(2,3,1)	1	0	0	0
(3,2,1)	0	1	0	0
(1,3,2)	1	0	0	1
...				

Propositionalization

CONACQ example *[Bessiere et al.]*

Monomials

Given

- a space of possible instances X
- an unknown target function f: X → Y
- a hypothesis space L containing functions X → Y   monomials conjunctions
- a set of examples E = { (x, f(x)) | x ∈ X }   pos only
- a loss function *loss*(h,E) → $\mathbb{R}$ error = 0

Find h ∈ L that minimizes *loss*(h,E)

Learning monomials

Represent each example by its set of literals

- $\{\neg X_1 , X_2 , \neg X_3 , X_4 , \neg X_5 \}$

Compute the intersection of all positive examples

- intersection = least general generalization

A cautious algorithm

Makes prudent generalizations

[Mitchell, ML textbook 97]

k-CNF

Given

- a space of possible instances X
- an unknown target function f: X → Y
- a hypothesis space L containing functions X → Y   k-CNF
- a set of examples E = { (x, f(x)) | x ∈ X }   pos only
- a loss function *loss*(h,E) → ℝ

Find h ∈ L that minimizes *loss*(h,E)

Learning k-CNF

Naive Algorithm *[Valliant CACM 84]*

- Let S be the set of all clauses with k literals
- for each positive example e
    - for all clauses s in S
        - if e does not satisfy s then remove s from S

polynomial (for fixed k) -- PAC-learnable

Where do the examples come from ?

Unkown probability distribution *P* is assumed on X

The examples in E are drawn at random according to *P*

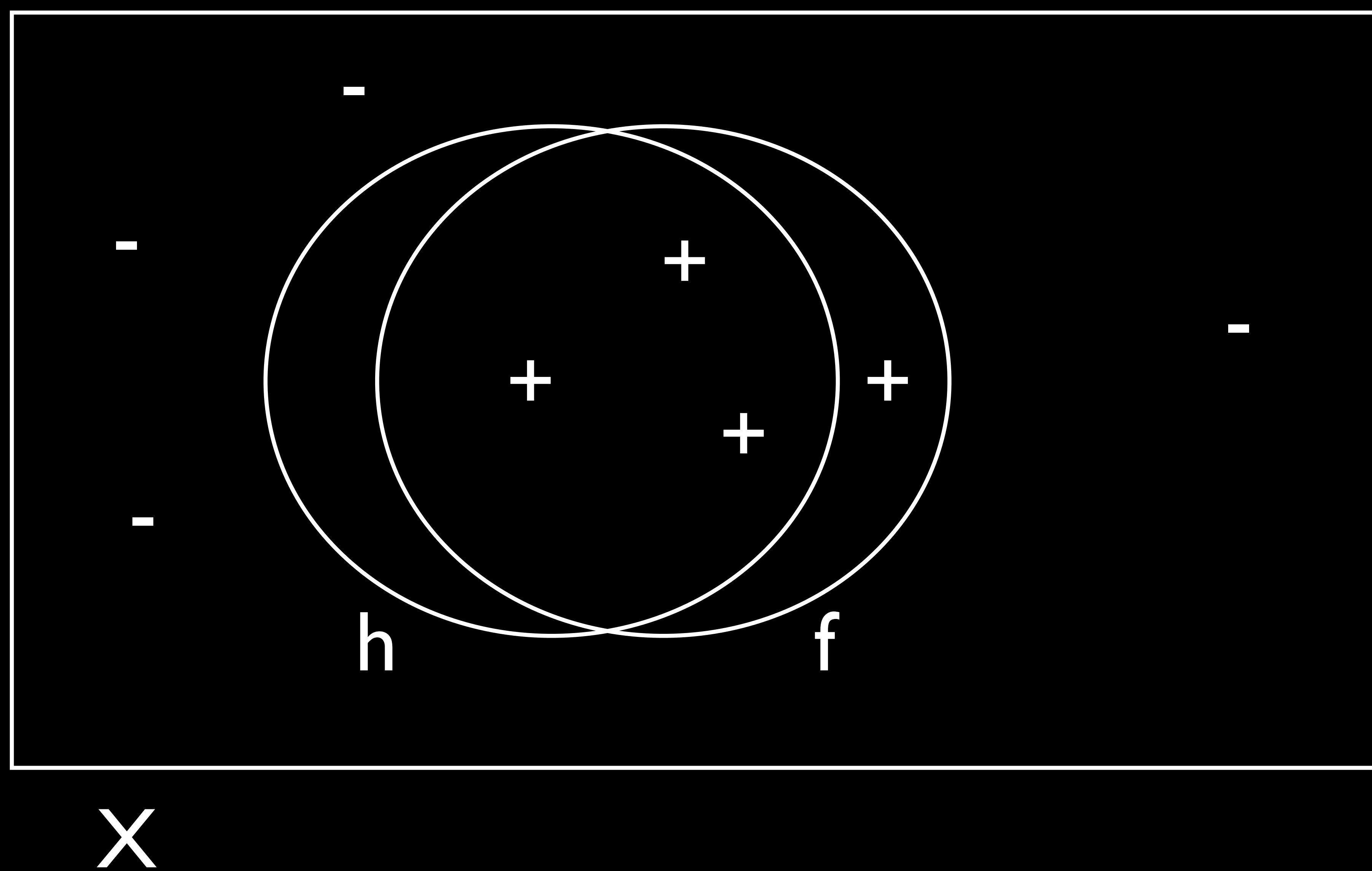
The i.i.d. assumption:

identically and independently distributed

(often does not hold for network / relational data)

Interpretation

Probability Distribution *P*



Classification Revisited

Make predictions about *unseen* data

$loss_l(h,E) = |\ \{e \mid e \in E, h(e) \neq f(e)\}\ |\ /\ |E|$

$= \text{training set error}$

$loss_t(h,X) = P\ (\{e \mid e \in X, h(e) \neq f(e)\})$

$= \text{true error}$

Formal Frameworks Exist

Probably Approximately Correct learning (PAC)

requires that learner finds with high probability approximately correct hypotheses

So, $P(\ loss_t(h,X) < \varepsilon) > 1-\delta$

Typically combined with complexity requirements

- sample complexity: number of examples
- computational complexity

Valliant proved polynomial PAC-learnability (fixed k)

Learning (k)-CNF

Alternative algorithm using Item-Set Mining principles

- minimum frequency = 100%
- clauses are disjunctions; itemsets conjunctions
- monotonicity property :
  - if e satisfies clause C then e also satisfies C U { lit }
  - interest in smallest clauses that satisfy 100% freq.
- frequency({ }) = 0, so refinement needed as for item-sets
- find upper border ...

DNF / rule learning

Given

- a space of possible instances X
- an unknown target function f: X → Y
- a hypothesis space L containing functions X → Y DNF
- a set of examples E = { (x, f(x)) | x ∈ X } pospos and neg
- a loss function *loss*(h,E) → ℝ error need not be 0

Find h ∈ L that minimizes *loss*(h,E)

Rule learning

Learning from Positives and Negatives

Learn a formula in Disjunctive Normal Form

Rule learning algorithms (machine learning)

Similar issues to pattern set mining (data mining perspective)

Rule learning is often heuristic

Set-covering algorithm

- repeatedly search for one rule (conjunction) that covers many positives and no negative
- discard covered positive examples and repeat

[Fuernkranz, AI Review 99, book 2010/11]

Asking Queries
Active Learning

Provide the learner with the opportunity to ask questions

Let T be the (unknown) target theory

- Does x satisfy T ? (membership)
- Does T |= X ? (subset)
- Does X |= T ? (superset)
- Are T and X logically equivalent ? (equivalence)
- ...

The oracle has to provide a counter-example in case the answer is negative *[Angluin, MLJournal 88]*

How can we use this?

Reconsider learning monomials (cf. *[Mitchell],* Conacq *[Bessiere et al]*)

Current hypothesis / conjunction

- $\{\neg X_1, X_2, \neg X_3, X_4, \neg X_5\}$
- generate example $\{X_1, X_2, \neg X_3, X_4, \neg X_5\}$
- if positive, delete X_1 , if negative, keep
- only n+1 questions needed to converge on unique solution (mistake bound)

Very interesting polynomial time algorithms for learning horn sentences *[Angluin et al. MLJ 92; Frazier and Pitt, ICML 93]* by asking queries

Generalizations

From propositional logic to first order logic

- Inductive Logic Programming

From ILP to Equation Discovery

From hard to soft constraints

- weighted MAX-SAT
- probabilistic models

Learning preferences

Inductive Logic Programming

Instead of learning propositional formulae, learn first order formulae

Usually (definite) clausal logic

Generalizations of many algorithms exist

Rule learning, decision tree learning

Clausal discovery *[De Raedt MLJ 97, De Raedt AIJ 94]*

- generalizes k-CNF of Valliant to first order case
- enumeration process as for k-CNF with border ...

Clausal Discovery in ILP

train(utrecht, 8, 8, denbosch) ←
train(maastricht, 8, 10, weert) ←
train(utrecht, 9, 8, denbosch) ←
train(maastricht, 9, 10, weert) ←
train(utrecht, 8, 13, eindhoven) ←
train(utrecht, 8, 43, eindhoven) ←
train(utrecht, 9, 13, eindhoven) ←
train(utrecht, 9, 43, eindhoven) ←

train(tilburg, 8, 10, tilburg) ←
train(utrecht, 8, 25, denbosch) ←
train(tilburg, 9, 10, tilburg) ←
train(utrecht, 9, 25, denbosch) ←
train(tilburg, 8, 17, eindhoven) ←
train(tilburg, 8, 47, eindhoven) ←
train(tilburg, 9, 17, eindhoven) ←
train(tilburg, 9, 47, eindhoven) ←

From1 = From2 ← train(From1, Hour1, Min, To), train(From2, Hour2, Min, To)

Inducing constraints that hold in data points
here functional dependencies
[De Raedt 97 MLJ, Flach AIComm 99,
Abdennaher CP 00, Lopez et al ICTAI 10, ...]

Equation Discovery

Instead of learning clauses, learn equations *[Dzeroski and Todorovski, Langley and Bridewell]*.

As Valiant's algorithm

- generate and test candidate equations, e.g., ax + byz = c
  - fit parameters using regression
- possibly compute values for additional variables (partial derivatives w.r.t. time, etc.)
- include a grammar to specify "legal equations" (bias)

Ecological Modeling

Table 1
Variables used in the NPPc portion of the CASA model

NPPc is the net production of carbon by terrestrial plants at a site
E is the photosynthetic efficiency at a site after factoring various sources of stress
T1 is a temperature stress factor ($0 < T1 < 1$) for cold weather
T2 is a temperature stress factor ($0 < T2 < 1$), nearly Gaussian in form but falling off more quickly at higher temperatures
W is a water stress factor ($0.5 < W < 1$)
topt is the average temperature for the month at which *fas_ndvi* takes on its maximum value at a site
tempc is the average temperature at a site for a given month
eet is the estimated evapotranspiration (water loss due to evaporation and transpiration) at a site
PET is the potential evapotranspiration (water loss due to evaporation and transpiration given an unlimited water supply) at a site
pet_tw_m is a component of potential evapotranspiration that takes into account the latitude, time of year, and days in the month
A is a polynomial function of the annual heat index at a site
ahi is an annual heat index that takes the time of year into account
fas_ndvi is the relative greenness as measured from space
IPAR is the energy intercepted from the sun after factoring in the time of year and days in the month
FPAR_FAS is the fraction of energy intercepted from the sun that is absorbed photosynthetically after factoring in vegetation type
monthly_solar is the average radiation incoming for a given month at a site
SOL_CONV is 0.0864 times the number of days in each month

$$NPPc = \max(0, E \cdot IPAR)$$
$$E = 0.312 \cdot T1^{1.36} \cdot T2^{0.728} \cdot W^{0}$$
$$T1 = 3.65 - 0.992 \cdot topt + 0.137 \cdot topt^2 - 0.00679 \cdot topt^3 + 0.000111 \cdot topt^4$$
$$T2 = 0.818/((1 + \exp(0.0521 \cdot (TDIFF - 10))) \cdot (1 + \exp(0 \cdot (-TDIFF - 10))))$$
$$TDIFF = topt - tempc$$
$$W = 0.5 + 0.5 \cdot eet/PET$$
$$PET = 1.6 \cdot (10 \cdot \max(tempc, 0)/ahi)^{A} \cdot pet\_tw\_m$$
$$A = 0.000000675 \cdot ahi^3 - 0.0000771 \cdot ahi^2 + 0.01792 \cdot ahi + 0.49239$$
$$IPAR = FPAR\_FAS \cdot monthly\_solar \cdot SOL\_CONV \cdot 0.5$$
$$FPAR\_FAS = \min((SR\_FAS - 1.08)/srdiff, 0.95)$$
$$SR\_FAS = (1 + fas\_ndvi/750)/(1 - fas\_ndvi/750)$$
$$SOL\_CONV = 0.0864 \cdot days\_per\_month$$

Using equation discovery to revise an Earth ecosystem model of the carbon net production

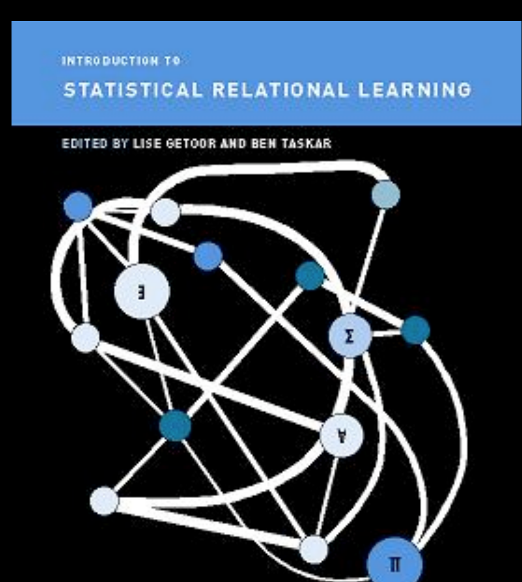
Ljupčo Todorovski [a,*], Sašo Džeroski [a],
Pat Langley [b], Christopher Potter [c]

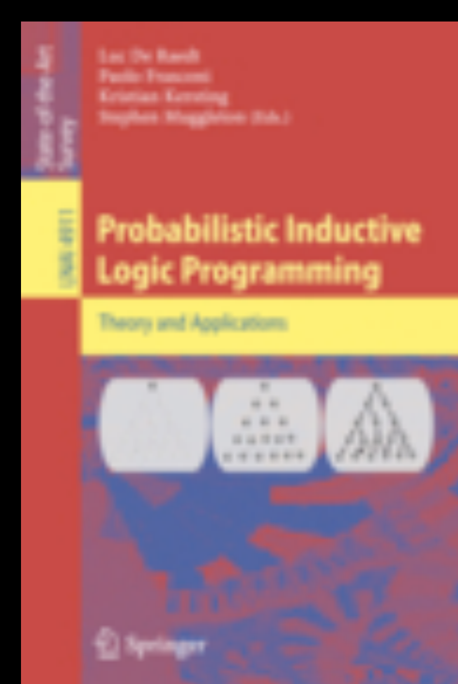
Learning Soft Constraints

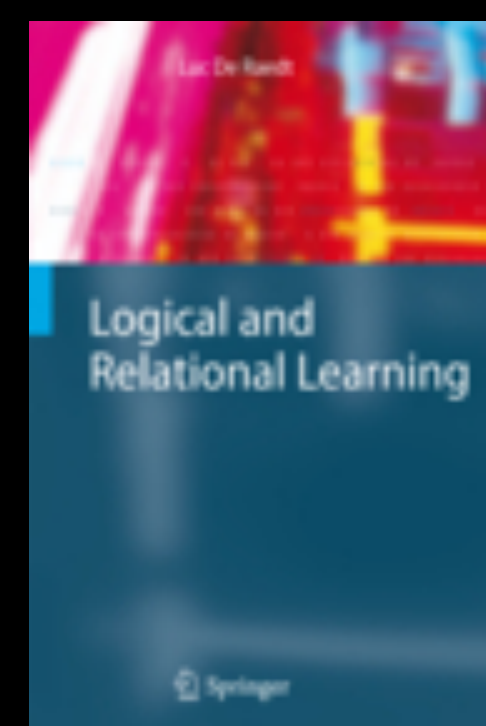
Let us look at weighted MAX-SAT problems

Quite popular today in Statistical Relational Learning

- combining first order logic, machine learning and uncertainty
- One example is Markov Logic, many others exist







Factors and Logic

- Propositional atoms are binary (0-1) variables.
- A joint instantiation of all atoms/variables satisfying a propositional formula is a *model* of that formula.
- If A and B are the only propositions in our language then $A, \neg A \vee B$ has only one model.

```
A |                 A | B |                 A | B |
- | -               - | - | -               - | - | -
0 | 0       *       0 | 0 | 1       =       0 | 0 | 0
1 | 1               0 | 1 | 1               0 | 1 | 0
                    1 | 0 | 0               1 | 0 | 0
                    1 | 1 | 1               1 | 1 | 1
```

Slide James Cussens

Generalizing Propositional Logic

- Allow arbitrary non-negative values in the factors.
- Allow variables to have more than 2 values.

A	
0	4
1	6

*

A	B	
0	0	5
0	1	5
1	0	0
1	1	7

=

A	B	
0	0	20
0	1	20
1	0	0
1	1	42

Dividing by a normalising constant Z defines a probability distribution over full joint instantiations (when $Z > 0$). Here $Z = 20 + 20 + 0 + 42 = 82$.

Slide James Cussens

Weighted Clauses

$\infty : A$ and $2 : \neg A \vee B$

A	
0	0
1	1

$*$

A	B	
0	0	1
0	1	1
1	0	exp(-2)
1	1	1

$=$

A	B	
0	0	0
0	1	0
1	0	exp(-2)
1	1	1

Finding the most probable instantiation (highest weighted model) is the weighted MAX-SAT problem.

e^{-w} where w=weight of clause if clause not satisfied; weight = 0 otherwise

Slide James Cussens

weighted MAX-SAT

Markov Logic uses weighted (first order logic) clauses to represent a Markov Network

Interesting inference and learning problems

- Compute P(X|Y) ...   (CP-techniques can help, weighted model counting)
- Compute most likely state (MAX-SAT)
- Learn parameters (weights of clauses)
  - e.g., using gradient descent on likelihood
- Learn structure and parameters

[Domingos et al], related to [Rossi, Sperduti KR, JETAI etc]

Learning Probabilistic Models

Given

- a space of possible instances X
- an unknown target function P: X → Y   Y=[0,1]
- a hypothesis space L containing functions X → Y (graphical models)
- a set of examples E = { (x, _) | x ∈ X }   generative
- a loss function *loss*(h,E) → $\mathbb{R}$ $\prod_{e \in E} P(e|h)$

Find h ∈ L that minimizes *loss*(h,E)   maximize likelihood

generative

Parameter Estimation

incomplete data set

A1	A2	A3	A4	A5	A6
true	true	?	true	false	false
?	true	?	?	false	false
...	...	...	...	...	...
true	false	?	false	true	?

states of some random variables are missing
E.g. medical diagnosis

Parameter Estimation

incomplete data set

A1	A2	A3	A4	A5	A6
true	true	?	true	false	false
?	true	?	?	false	false
...	...	...	...	...	...
true	false	?	false	true	?

hidden/latent

states of some random variables are missing

E.g. medical diagnosis

missing value

Preference learning

Problem with previous approach

- hard to sample examples from probability distribution in CP context; or to give examples with target probability

A hot topic today in ML, many variations exist, cf. *[Furnkranz and Eykemuller, 10, book & tutorial -- videolectures]*

Two main settings

- learning object preferences (model acquisition)
- learning label preferences (portfolio's)

Object Preferences

Given

- a space of possible instances X
- an unknown ranking function r(.), given O⊆X, rank instances in O
- a hypothesis space L containing ranking functions
- a set of examples E = { (x > y) | x,y ∈ X }
- a loss function *loss*(h,E) → $\mathbb{R}$

Find h ∈ L that minimizes *loss*(h,E)

Possible approaches

Explicit relation learning

- Learn a relation $Q(x,y)$ from examples $x < y$
- Determine $r(O)$ as the ordering that is maximally consistent with Q

Learn latent utility function

- an unknown utility function $f: X \to \mathbb{R}$
- examples only impose constraints on f
    - values of f not known

Label Preferences

Given

- a space of possible instances X
- a set of labels Y = $\{Y_1, \ldots, Y_n\}$
- an unknown target function f(x) = permutation of Y
- a set of examples E = $\{ (x , \{ Y_i > Y_j \})\}$
- a loss function *loss*(h,E) $\rightarrow \mathbb{R}$

Find h $\in$ L that minimizes *loss*(h,E)

Possible approaches

Learn set of relations for each $Y_i > Y_j$

Learn latent utility function for each label Y_i

An unknown utility function $f_i : X \to \mathbb{R}$

- examples only impose constraints on f_i :
  - values of f not known

Summary

The learning of CSPs is possible, so let's do it

Many settings exist

- data, hypothesis language, active, soft constraints, preference learning, etc

Still we did not touch upon

- bayesian and statistical learning methods

One interesting approach that learns MAX-SAT and MAX-SMT by asking preference questions and using statistical learning techniques

. Campigotto, A. Passerini and R. Battiti, Lion 10 workshop

Further reading -- Encyclopedia of Machine Learning

References [Part II]

- Tom Mitchell, Machine Learning, Mc GrawHill,1997.
- Encyclopedia of Machine Learning, Springer, 2010.
- Johannes Fürnkranz and Eyke Hüllermeier, Preference Learning, in: Encyclopedia of Machine Learning, Springer-Verlag, 2010. Also, edited book and videolectures !
- C. Bessiere, R. Coletta, F. Koriche, B. O'Sullivan, Acquiring Constraint Networks using a SAT-based Version Space Algorithm, Proceedings of AAAI'06, Nectar paper, Boston Massachusetts, July 2006.
- . Campigotto, A. Passerini and R. Battiti, Lion 10 workshop
- Alessandro Biso, Francesca Rossi, Alessandro Sperduti: Experimental Results on Learning Soft Constraints. KR 2000: 435-444 and later papers
- Pedro Domingos, Daniel Lowd: Markov Logic: An Interface Layer for Artificial Intelligence Morgan & Claypool Publishers 2009
- Dzeroski, S. and Todorovski, L. (1995) Discovering dynamics: From inductive logic programming to machine discovery. Journal of Intelligent Information Systems, 4: 89-108. and later papers
- Luc De Raedt, Luc Dehaspe: Clausal Discovery. Machine Learning 26(2-3): 99-146 (1997)
- Luc De Raedt, Logical and Relational Learning, Springer, 2008.

References [Part II]

- Dana Angluin: Queries and Concept Learning. Machine Learning 2(4): 319-342 (1987)
- Dana Angluin, Michael Frazier, Leonard Pitt: Learning Conjunctions of Horn Clauses. Machine Learning 9: 147-164 (1992)
- Michael Frazier, Leonard Pitt: Learning From Entailment: An Application to Propositional Horn Sentences. ICML 1993: 120-127
- Peter A. Flach, Iztok Savnik: Database Dependency Discovery: A Machine Learning Approach. AI Commun. 12(3): 139-160 (1999)
- Slim Abdennadher, Christophe Rigotti: Automatic Generation of Propagation Rules for Finite Domains. CP 2000: 18-34
- Arnaud Lallouet, Matthieu Lopez, Lionel Martin, Christel Vrain: On Learning Constraint Problems. ICTAI (1) 2010: 45-52

References [Part III]

- Rakesh Agrawal, Heikki Mannila, Ramakrishnan Srikant, Hannu Toivonen, A. Inkeri Verkamo: Fast Discovery of Association Rules. Advances in Knowledge Discovery and Data Mining 1996: 307-328
- Rakesh Agrawal, Ramakrishnan Srikant: Mining Sequential Patterns. ICDE 1995: 3-14
- Roberto J. Bayardo Jr.: Efficiently Mining Long Patterns from Databases. SIGMOD Conference 1998: 85-93
- Sugato Basu, Ian Davidson. Tutorial on clustering with constraints at http://www.cs.ucdavis.edu/~davidson/
- Christian Bessiere, Emmanuel Hebrard, Barry O'Sullivan: Minimising Decision Tree Size as Combinatorial Optimisation. CP 2009: 173-187
- Jean-Phillipe Métivier, Patrice Boizumault, Bruno Crémilleux, Mehdi Khiari, Samir Loudni. A constraint-based language for declarative pattern discovery. Workshop on Declarative Pattern Mining, 2011.
- Luc Dehaspe, Luc De Raedt: Mining Association Rules in Multiple Relations. ILP 1997: 125-132
- Luc De Raedt, Albrecht Zimmermann: Constraint-Based Pattern Set Mining. SDM 2007
- Luc De Raedt, Tias Guns, Siegfried Nijssen: Constraint programming for itemset mining. KDD 2008: 204-212

References [Part IV]

- Saher Esmeir, Shaul Markovitch: Anytime Induction of Cost-sensitive Trees. NIPS 2007
- Johannes Fischer, Volker Heun, Stefan Kramer: Optimal String Mining Under Frequency Constraints. PKDD 2006: 139-150
- Sean Gilpin, Ian Davidson: Incorporating SAT solvers into hierarchical clustering algorithms: an efficient and flexible approach. KDD 2011: 1136-1144
- Tias Guns, Siegfried Nijssen, Albrecht Zimmermann, Luc De Raedt. Declarative Heuristic Search for Pattern Set Mining. Workshop on Declarative Pattern Mining, 2011.
- T. Imielinski and H. Mannila. A database perspective on knowledge discovery. Communications of the ACM, 3911:58 64, November 1996.
- Akihiro Inokuchi, Takashi Washio, Hiroshi Motoda: Complete Mining of Frequent Patterns from Graphs: Mining Graph Data. Machine Learning 50(3): 321-354 (2003)
- Siegfried Nijssen, Élisa Fromont: Optimal constraint-based decision tree induction from itemset lattices. Data Min. Knowl. Discov. 21(1): 9-51 (2010)
- Siegfried Nijssen, Élisa Fromont: Mining optimal decision trees from itemset lattices. KDD 2007: 530-539
- Siegfried Nijssen, Tias Guns, Luc De Raedt: Correlated itemset mining in ROC space: a constraint programming approach. KDD 2009: 647-656
- Nicolas Pasquier, Yves Bastide, Rafik Taouil, Lotfi Lakhal: Discovering Frequent Closed Itemsets for Association Rules. ICDT 1999: 398-416
- Mohammed Javeed Zaki: Efficiently mining frequent trees in a forest. KDD 2002: 71-80

Thank you