Data Mining using Logic

Tightening the Gap between
Logic and Specialised Algorithms

Siegfried Nijssen

`snijssen@liacs.nl`

March 7, 2001

Overview

- Why first order logic?
- What is ILP?
- Frequent Itemsets
- Decision Trees
- Conclusions

Why first order logic?

- most knowledge discovery algorithms assume that data is in *one* table.

- much databases do not consist of one table.

- translation from multiple tables to one table not always sufficiently possible.

Why first order logic?

- Example 1: financial dataset. 1 transaction $\Rightarrow$ several clients.

Transaction

Date	Amount	Type	Account
010199	100	cw	123023
010199	10	cc	232390
030299	23	tb	348930
230299	345	fb	394813
030499	35	fb	348341

Client

Client	Address	Sex
342	167	F
645	184	M
367	123	F
345	153	M
543	123	M

Disposition

Account	Type	Client
123023	owner	342
348930	disp	543
348930	owner	367
232390	disp	645
394813	disp	345
348341	owner	345

Why first order logic?

- Solution 1: join the tables.

Transaction			Disposition		Client		
Date	**Amount**	**Type**	**Account**	**Type**	**Client**	**Address**	**Sex**
010199	100	cw	123023	owner	342	167	F
010199	10	cc	232390	disp	645	184	M
030299	23	tb	348930	owner	367	123	F
030299	23	tb	348930	disp	543	123	M
230299	345	fb	394813	disp	345	153	M
030499	35	fb	348341	owner	345	153	M

$\Rightarrow$ duplicated records.

Why first order logic?

- Solution 2: new columns.

Transaction			Disposition1		Client1			Disposition2		Client2	
Date	Amount	Type	Account	Type	Client	Address	Sex	Type	Client	Address	Sex
010199	100	cw	123023	owner	342	167	F				
010199	10	cc	232390	disp	645	184	M				
030299	23	tb	348930	owner	367	123	F	disp	543	123	M
230299	345	fb	394813	disp	345	153	M				
030499	35	fb	348341	owner	345	153	M				

$\Rightarrow$ empty records.

Why first order logic?

- Example 2: financial dataset. 1 client $\Rightarrow$ several transactions.

Transaction

Date	Amount	Type	Account
010199	100	cw	123023
010199	10	cc	232390
030299	23	tb	348930
230299	345	fb	394813
030499	35	fb	348341

Client

Client	Address	Sex
342	167	F
645	184	M
367	123	F
345	153	M
543	123	M

Disposition

Account	Type	Client
123023	owner	342
348930	disp	543
348930	owner	367
232390	disp	645
394813	disp	345
348341	owner	345

Why first order logic?

- Solution 3: summarize attributes.

Client

Client	Address	Sex	AvgAmount	# Account
342	167	F	100	1
645	184	M	10	1
367	123	F	23	1
345	153	M	190	2
543	123	M	23	1

$\Rightarrow$ information lost.

Why first order logic?

- All previous solutions have disadvantages.

- Why not use logic?
  - logic can express relations;

    $$\forall C \in Client : \exists A \in Account, T \in Type : disposition(C, A, T)$$

  - no need to change tables.

What is ILP?

- ILP = **I**nductive **L**ogic **P**rogramming.
  = the automatic generation of logic programs.

- Two flavours:

  - *predicate learners*,
    which learn from a set of examples a predictive program that defines a predicate.

  - *non-monotonic learners*,
    which create a descriptive set of formulas that expresses relations between examples.

What is ILP?

- Example: predicate learner for `daughter`.

  Given background knowledge:

```
parent(wilhemina,juliana), parent(juliana,beatrix),
parent(bernard,beatrix), parent(claus,wa), parent(beatrix,wa),
male(claus),male(wa),female(wilhemina),female(beatrix),
female(juliana), male(bernard).
```

  And examples:

```
daughter(beatrix,bernard),daughter(juliana,wilhemina).
```

What is ILP?

Conclude that:

$$\forall X, Y : daughter(X, Y) \leftarrow (parent(Y, X), female(X))$$

- Example: non-monotonic learner.

  Given background knowledge:

```
b(1,pizza). livesat(1,leiden). worksat(1,leiden).
b(2,chocolate). livesat(2,denHaag). worksat(2,leiden).
b(3,chocolate). livesat(3,denHaag). worksat(3,denHaag).
```

What is ILP?

Conclude that:

$$\forall Y : livesat(X,Y) \rightarrow worksat(X,Y)$$
$$\forall Y : worksat(X,Y) \rightarrow livesat(X,Y)$$
$$b(X, chocolate) \wedge livesat(X, denHaag)$$
$$\vdots$$

- Differences:
  - ▶ not only rules with $\forall$ and $\rightarrow$.
  - ▶ rules with $\forall$ and $\rightarrow$ need not be complete description.

Frequent Itemsets

- 'Set of items frequently bought together':

  cheese, bread, butter

- Equivalent in logic:

$$b(X, cheese) \wedge b(X, bread) \wedge b(X, butter)$$

- *Frequent Queries*: add variables and ∃s:

$$\exists Y : livesat(X, Y) \wedge worskat(X, Y)$$

Frequent Itemsets

- For both queries and itemsets:
  - ▶ the *support* can be counted;
  - ▶ the support of a subset is equal or higher;
  - ▶ a level-wise search starting with the smallest set can be performed.

- Apriori: frequent itemsets; Warmr: frequent queries.

Frequent Itemsets

- Problem with Warmr: inefficient! Causes:
  - ▶ relies on Prolog to count queries;
  - ▶ does not exploit overlap between queries;
  - ▶ uses an NP complete operator to check *equivalence*.

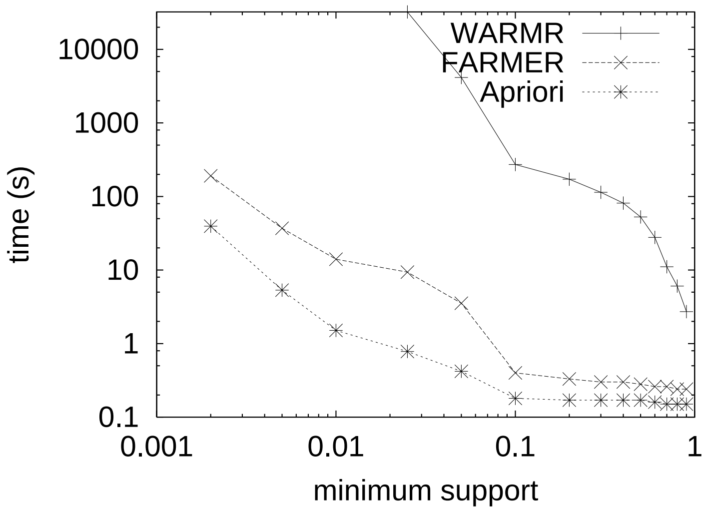
$$b(X,P_1), cheap(P_1), b(X,P_2), P_2 \neq P_1, cheap(P_2), yellow(P_2).$$
$$b(X,P_1), cheap(P_1), yellow(P_1), b(X,P_2), P_2 \neq P_1, cheap(P_2).$$

Frequent Itemsets

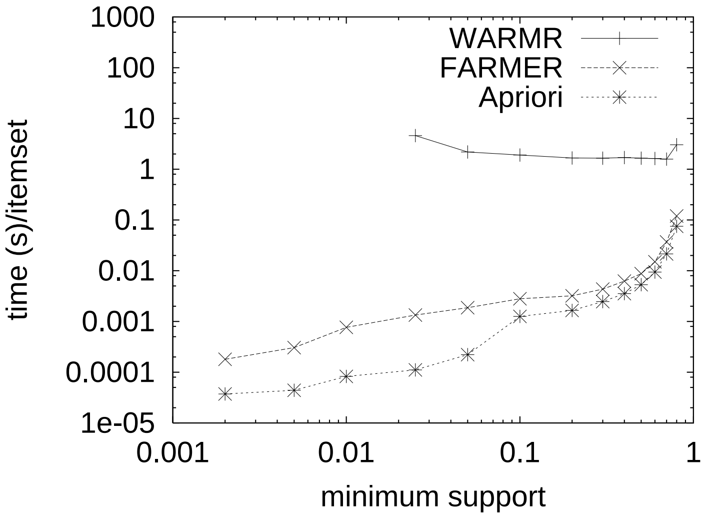
- New algorithm: Farmer:
  - does not rely on Prolog;
  - exploits overlap while counting and creating queries;
  - does not use an equivalence operator, relies on tree building procedure and some additional methods.

Frequent Itemsets

- Evaluation experiments.

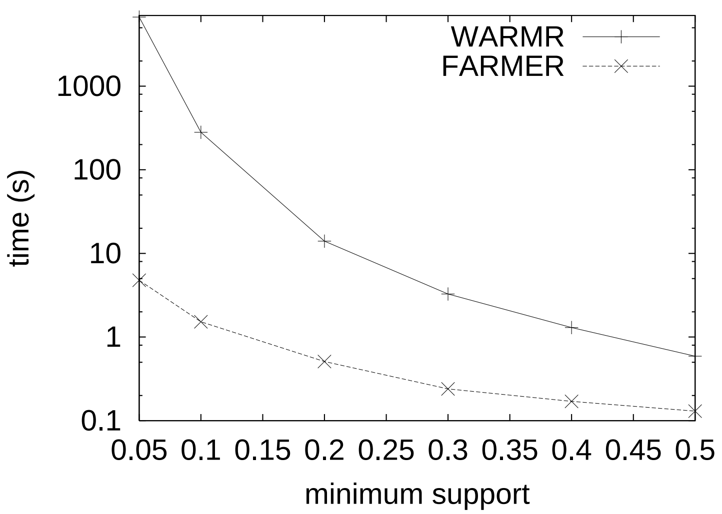


Itemsets Time

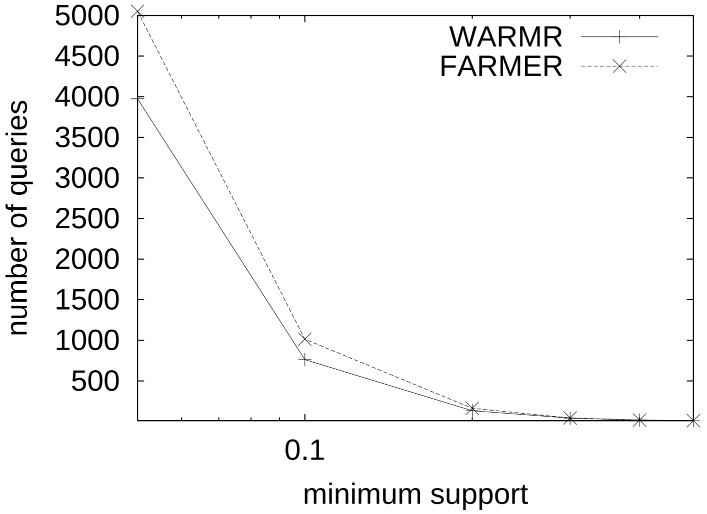


Time/Itemset

Frequent Itemsets



Queries Time



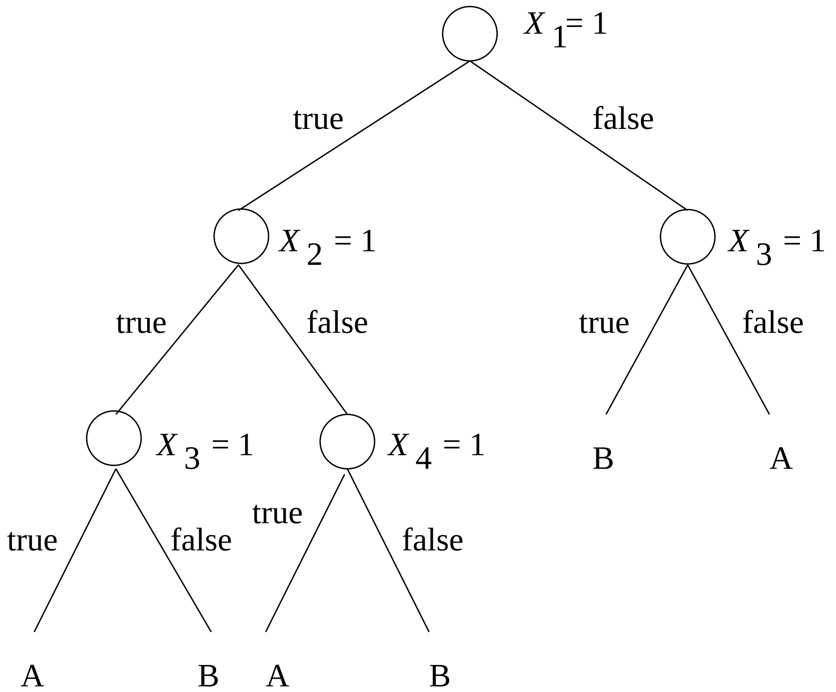
Queries nr

Frequent Itemsets

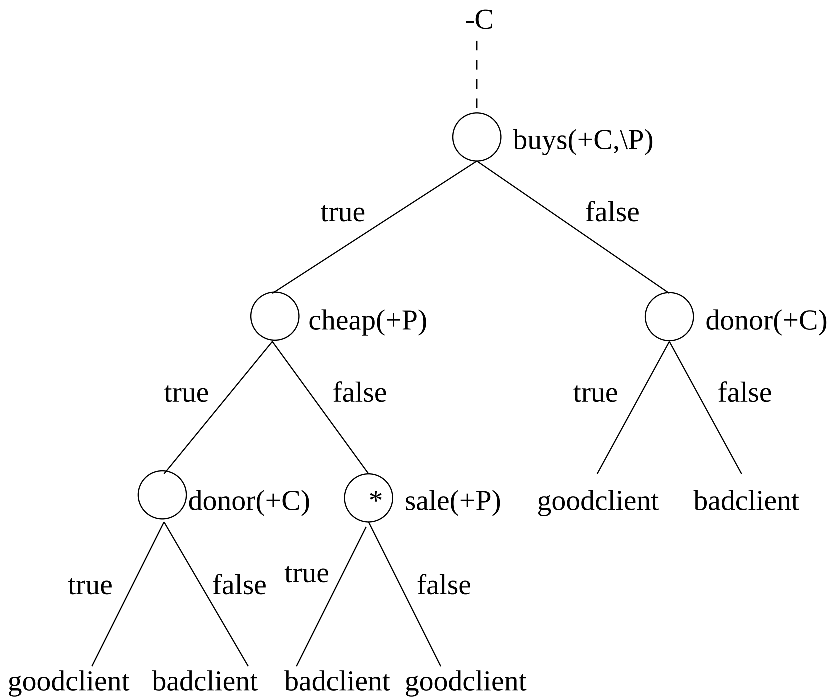
- Financial experiments. Some results (10% threshold):

  "heavily urbanized districts with high average salaries generate more than 10% of the transactions, where it is remarkable that most of the accounts associated with these transactions, have permanent household orders."
  "12% of the transactions is generated by an account for which there is a disponent and an owner living in the same district."
  "12% of the transactions is generated by an account for which there is a male and a female disponent living in the same district."
  A combination of the lather two does not occur.

Decision Trees



‘Normal’



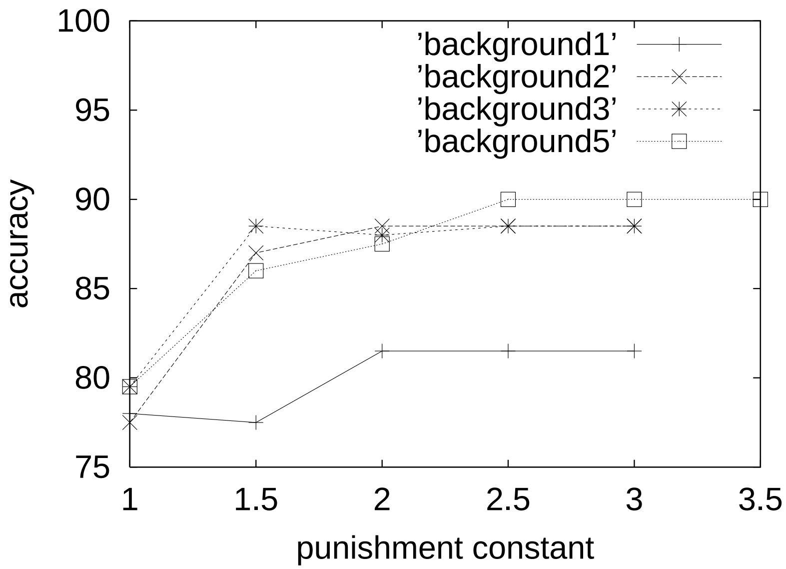
Logic Decision Tree

Decision Trees

- C4.5: ‘normal’ decision trees; Tilde: logic decision trees.
- ‘Problems’ with Tilde:
  - relies on Prolog to split examples in leafs;
  - does not exploit overlap between paths;
  - peculiar heuristics:
    - continuous attributes use worse heuristic from C4.5;
    - no punishment for overfitting.

Decision Trees

- Evaluation experiments.
  - Overfitting punishment



  - No convincing results in other experiments.

Decision Trees

- Financial experiments. Some results:

  "loans with a second disponent are always paid back."
  "loans of clients with permanent household orders are more frequently paid back."
  "people with high loan amounts perform credit card transactions more often."
  "in December the number of cash transactions is larger."
  "all 144092 payments of amounts between 14 and 14.9 are in cash."

Conclusions

- scale-up has reasonably succeeded, with an ILP algorithm testing 1.000.000 transactions.

- however too many rules/nodes, pruning is necessary.