Naive Bayesian Classification of Structured Data

Peter Flach, U. of Bristol & Nicolas Lachiche, U. of Strasbourg

ILP'99, ILP'02, ICML'03, Machine Learning 57(3):233-269, 2004

- Motivation and background
- 1BC
- 1BC2
- Comparison

Motivation: upgrading naive Bayes

- Bayesian classifier:

$$\arg\max_c P(c \mid d) = \arg\max_c P(d \mid c) P(c)$$

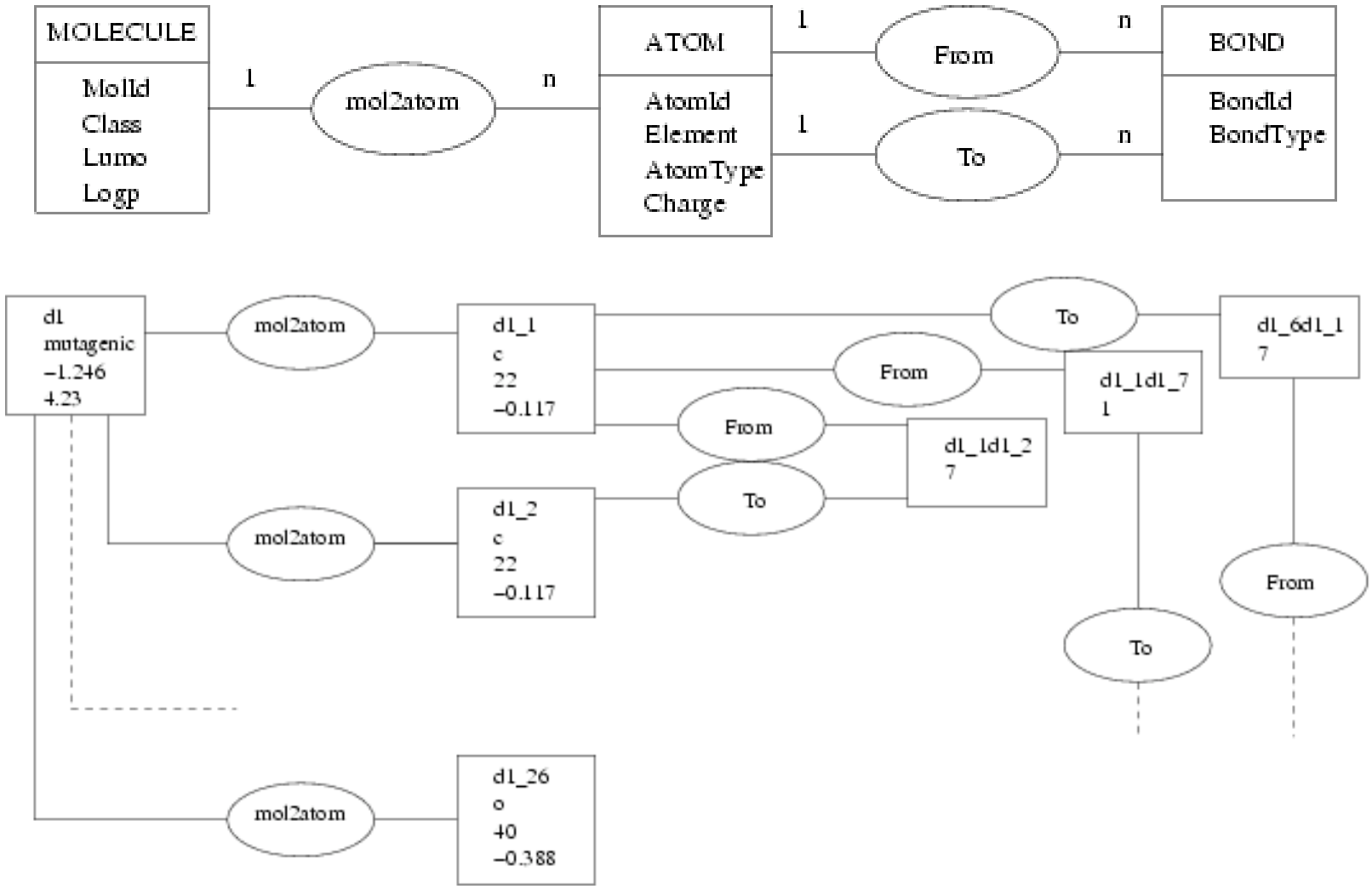
- Propositional naive Bayes assumption:

$$\arg\max_c P(\bigwedge_i A_i = a_i \mid c) P(c)$$

$$\arg\max_c P(c) \prod_i P(A_i = a_i \mid c)$$

- 1 to n: P(d|c)?

E-R model


MOLECULE
MolId
Class
Lumo
Logp
1
mol2atom
n
ATOM
AtomId
Element
AtomType
Charge
1
From
n
1
To
n
BOND
BondId
BondType
d1
mutagenic
−1.246
4.23
mol2atom
d1_1
c
22
−0.117
To
d1_6d1_1
7
From
d1_1d1_7
1
From
d1_1d1_2
7
To
mol2atom
d1_2
c
22
−0.117
From
To
mol2atom
d1_26
o
40
−0.388

ISP declarations

```
--INDIVIDUAL
mol 1 mol
--STRUCTURAL
mol2atom 2 1:mol *:atom
from 2 1:atom *:bond
to 1:atom *:bond
--PROPERTIES
class 2 mol #class
element 2 atom #element
atomtype 2 atom #type
charge 2 atom #charge
bondtype 2 bond #bondtype
```

Flattened representation

```
class(d1,mutagenic).
mol2atom(d1,d1_1).
mol2atom(d1,d1_2).
...
element(d1_1,c).
atomtype(d1_1,22).
charge(d1_1,-0.117).
...
from(d1_1,d1_1d1_2).
to(d1_2,d1_1d1_2).
bondtype(d1_1d1_2,7).
...
```

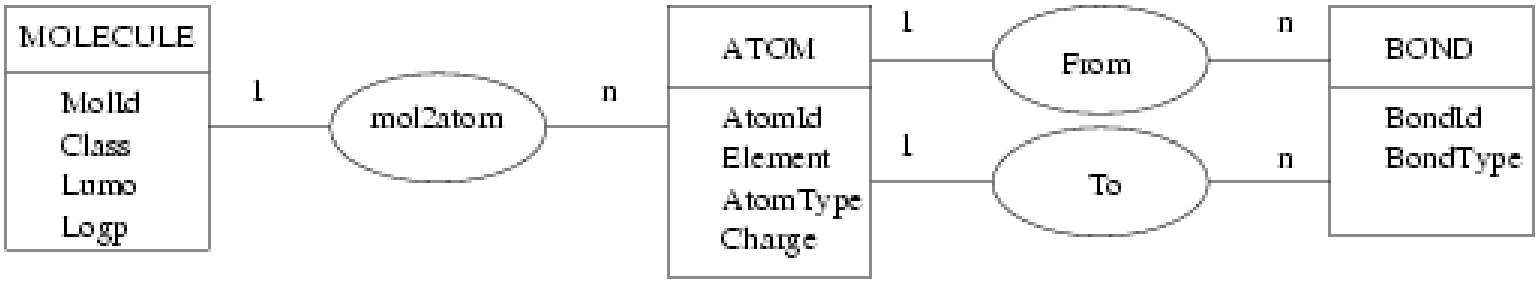
1BC

- Upgrades attributes to first-order features
- Recursively follows the type structure
  - either a property of an object O, or a first-order feature of an object O' related to O through a structural predicate
  - propositionalisation

First-order features

- First-order feature is conjunction of literals such that
  - one free global variable denoting the individual
  - each (binary) structural predicate introduces a new local variable and uses either the global variable or a local variable introduced by other structural predicates
  - properties do not introduce new variables
  - all variables are used at least once
  - bounded number of literals and variables

Elementary features



- Elementary features: one property per feature
  - lumo(M,-1.246)
  - mol2atom(M,A),element(A,c)
  - mol2atom(M,A),from(A,B),bondtype(B,7)

1BC: dynamic propositionalisation

- Learning

```
FOR each value f of each feature F
```

$$estP(f|c) = \frac{n(f \wedge c) + 1}{n(c) + \Phi}$$

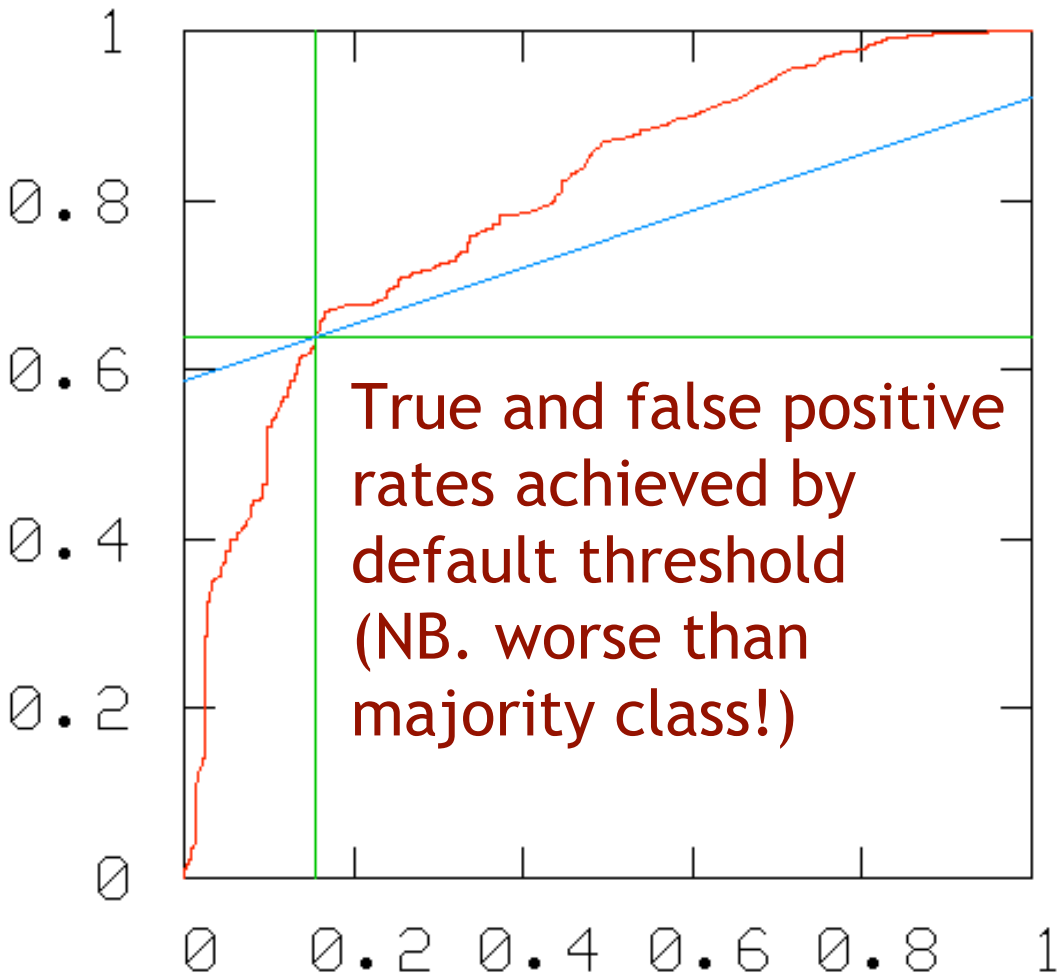
- Classification

```
getConditionalLikelihood(i) {
FOR each class c
    CL[c] = 1
FOR each feature F
    let f be the value of F for i
    FOR each class c
        CL[c] = CL[c] x estP(f|c)
RETURN CL}
```

Using likelihoods for classification

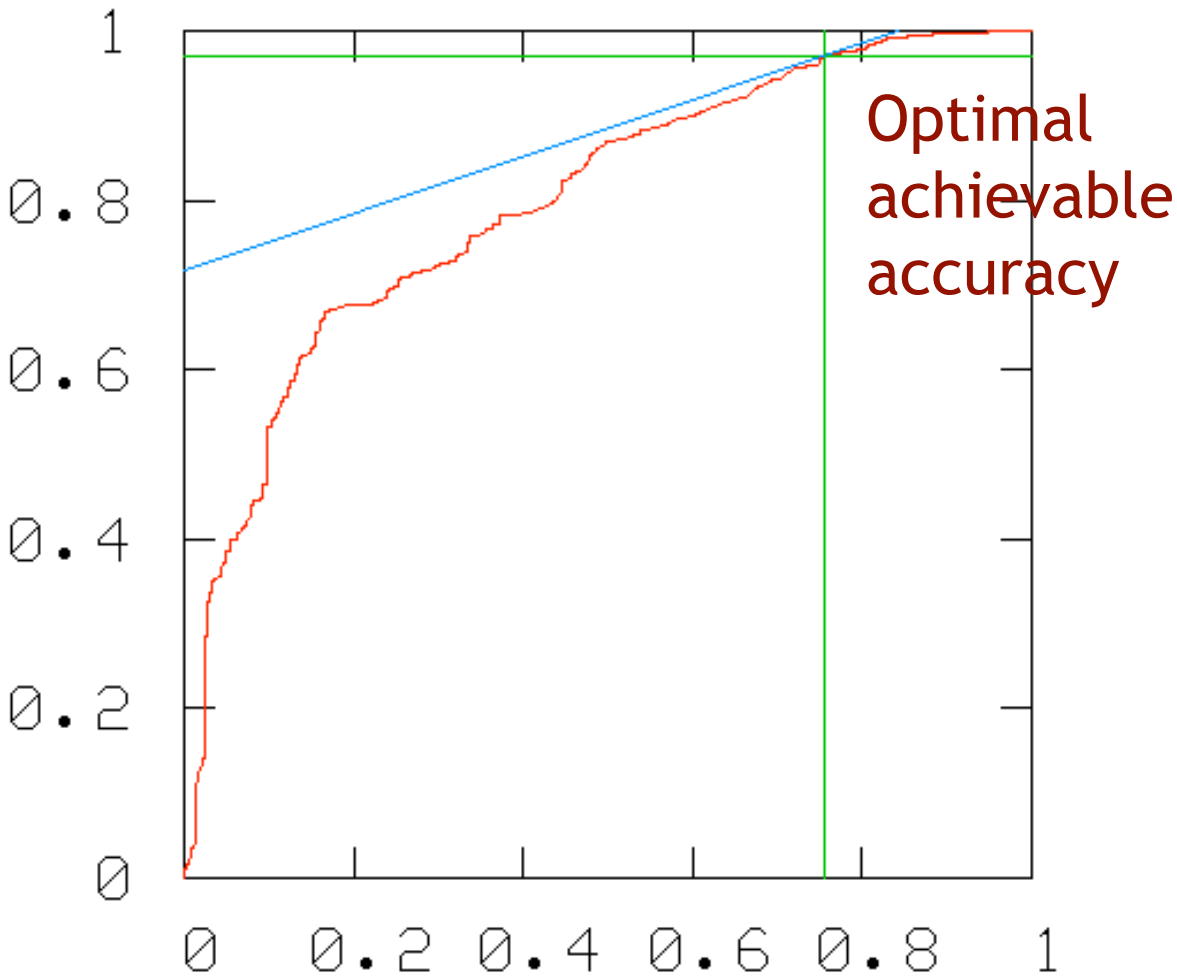
- Requires decision rule, e.g., setting a threshold on the likelihood ratio
  - Bayesian: predict positive if $\frac{P(x \mid +)}{P(x \mid -)} > \frac{Neg}{Pos}$
    - or threshold posterior: $\frac{P(x \mid +) \cdot Pos}{P(x \mid -) \cdot Neg} > 1$
- This only makes sense if the likelihood ratios are calibrated
  - not true for naïve Bayes because of unrealistic independence assumptions
  - i.e. ignore the prior, and learn the optimal decision threshold from data!

Uncalibrated threshold



True and false positive rates achieved by default threshold (NB. worse than majority class!)

Calibrated threshold

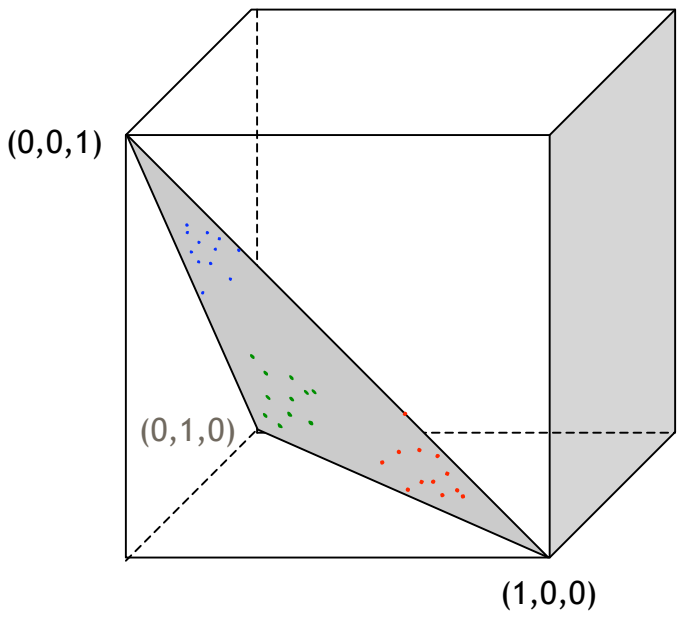


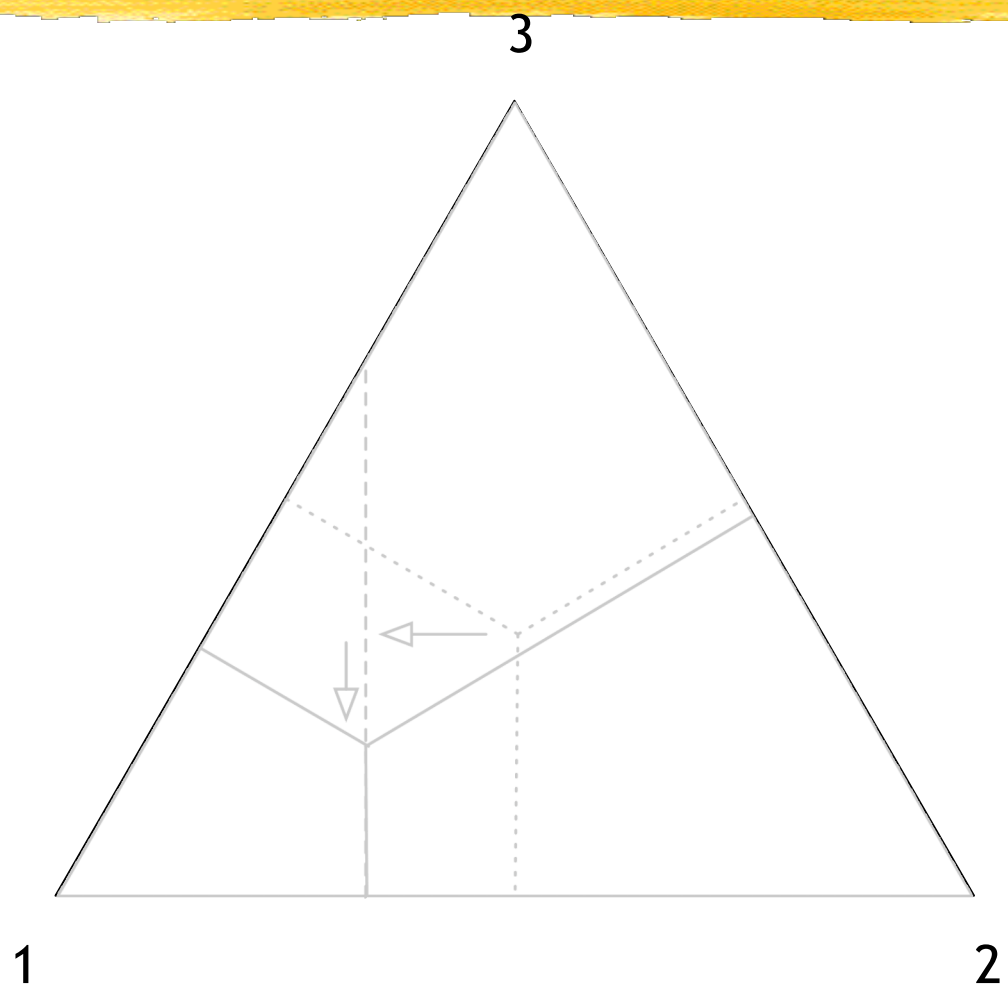
n>2 classes: a simple 1-vs-rest approach

- From thresholds to weights:
  - predict $\text{argmax}_c\ w_c\ P(x|c)$
  - NB. two-class thresholds are a special case:
    - $w_+\ P(x|+) > w_-\ P(x|-) \Leftrightarrow P(x|+)/P(x|-) > w_-/w_+$

- Setting the weights (Lachiche & Flach, ICML'03)
  - Assume an ordering on classes and set the weights in a greedy fashion
    - Set $w_1 = 1$
    - For classes c=2 to n
      - look for the best weight w_c according to the weights fixed so far for classes c'<c, using the two-class algorithm

Example: 3 classes





1BC2

- Upgrades the naive Bayes assumption from tuples to collections formed by one-to-many relationships
  - e.g., collection of atoms appearing in molecule
- Need to model probability distributions over collections
  - we considered lists, multisets and sets

What's wrong with bit-vectors?

- Represent a collection by a tuple of bits, one for each domain element
  - aka propositionalisation with existential features
- Only works for finite domains
- Defines probability of a collection in terms of its extension **and** its complement

Unigram list SLP

```
0.2: element(a).
0.3: element(b).
0.5: element(c).

0.28: list([]).
0.72: list([H|T]):-element(H),list(T).
```

- all permutations equi-probable

Unigram list SLP (alternative definition)

```
0.2*0.72: element(a).
0.3*0.72: element(b).
0.5*0.72: element(c).
0.28: element(stop).

list([H|T]):-
  element(H),
  ( H=stop     -> T=[]
  ; otherwise -> list(T)
  ).
```

- all permutations equi-probable

Bigram list SLP

```
0.05: pair(a,a).
0.10: pair(a,b).
...
0.15: pair(c,c).

0.28: listp([]).
0.20: listp([X]):-element(X).
0.52: listp([X,Y|T]):-pair(X,Y),listp([Y|T]).
```

- `[a,a,b,a]` and `[a,b,a,a]` equi-probable

1BC2 list distribution (unigrams)

- Alphabet $A = \{x_1, \ldots, x_n\}$ with prob.dis. P_A
  - probability of empty list is τ

$$P_{\mathrm{li}}([x_{j_1}, \ldots, x_{j_l}]) = \tau(1-\tau)^l \prod_{i=1}^{l} P_A(x_{j_i})$$

- Equivalently: alphabet $A' = \{\varepsilon, x_1, \ldots, x_n\}$
  - $P_{A'}(\varepsilon) = \tau$, $P_{A'}(x_i) = (1-\tau)P_A(x_i)$,

$$P_{\mathrm{li}}([x_{j_1}, \ldots, x_{j_l}]) = P_{A'}(\varepsilon) \prod_{i=1}^{l} P_{A'}(x_{j_i})$$

From lists to multisets

- Idea: view a multiset as the equivalence class of all lists consisting of its permutations
  - k_i denotes number of occurrences of x_i

$$P_{\mathrm{ms}}(\{\{x_{j_1},\ldots,x_{j_l}\}\}) = \frac{l!}{k_1!\ldots k_n!} P_{\mathrm{li}}([x_{j_1},\ldots,x_{j_l}]) = l!\,\tau \prod_{i=1}^{n} \frac{P_{A'}(x_i)^{k_i}}{k_i!}$$

- gives same results as list distribution for naïve Bayes classification since #permutations only depends on instance to be classified, not on class

From lists to sets

- Distribution over sets
  - sums up the probabilities of all lists containing each element of the set at least once, and no other element
  - only takes the elements occurring in the set into account, in contrast to bitvectors
  - necessarily exponential in the cardinality of the set
  - can be approximated by the list distribution for large domains

1BC2 set distribution

- Idea: view set S of cardinality l as equivalence class of lists of length at least l containing all and only elements of S
  - $P_{A'}(S)$ denotes $\sum_{x \in S} P_{A'}(x)$
  - cumulative prob. of lists of length at least l containing only elements from S is $f(S) = \sum_{j \geq l} \tau (P_{A'}(S))^j = \tau \frac{(P_{A'}(S))^l}{1 - P_{A'}(S)}$

$$P_{ss}(S) = \sum_{S' \subseteq S} (-P_{A'}(S'))^{l-l'} \times f(S')$$

  - $P_{SS}(S) \sim l!\, P_{li}(S)$ if $P_{A'}(S)$ is close to 0

1BC2: main approach

- Recursively follows the individual structure:
  - the estimated probability of an individual i is the product of
  - the estimated probability of its properties and
  - the estimated probability of the collection of objects i' related to i through a structural predicate

1BC2: Learning phase

- Estimate probabilities of each property for each class
  - e.g. carbon atoms belonging to mutagenic molecules
- Estimate probabilities of empty collections (MLE):

$$\tau = \frac{1}{1 + \bar{l}}$$

  - e.g. average number of atoms in molecules = 4
  - --> $\tau = 0.2$

getConditionalLikelihood(individual i)→ CL

```
FOR each class c, CL[c] = 1 /* initialisation */
FOR all properties prop of i except class
    Find parameter value v such that prop(i,v) is true
    For each class c, CL[c] = CL[c] x estP(v|c)
FOR all structural predicates struc involving i
    IF struc is functional given i THEN
        Find the related individual j
        CL' = getConditionalLikelihood(j)
        FOR each class c, CL[c] = CL[c] x CL'[c]
    ELSE /* non-determinate struc.pred.*/
        Find all related individuals J
        FOR each individual j of J,
            CL''[j] = getConditionalLikelihood(j)
        FOR each class c,
            CL[c] = CL[c] x composeP(CL'',tau,c)
RETURN CL
```

Comparison

	1BC	1BC2
Counts	Individual	Objects
Cardinalities	No	Yes
Repetitions	No	Yes
Properties	Satisfied and non-satisfied	Satisfied only
1 to n	Bitvector	Distributions over collections

Experimental evaluation

- Artificial datasets: 1BC2 outperforms 1BC
  - cardinality of collection
  - distribution of elements of collection
  - requires more data to estimate these probabilities reliably
- Benchmark ILP datasets
  - Both perform comparably to other ILP systems

Example: training and test data

- Positive examples
  - {(o,22),(c,14),(o,21),(o,22),(n,38),(h,9)}
  - {(h,9),(o,22),(o,22),(n,38),(h,9)}
  - {(o,21),(o,22),(c,14),(h,9)}
- Negative examples
  - {(c,14),(c,17)}
  - {(h,9),(o,22),(c,14)}
  - {(o,22)}
- Test individual
  - {(o,22),(c,17),(h,9),(h,9),(h,9)}

Example: conditional probabilities

	1BC		1BC2	
	positive	negative	positive	negative
mol2atom(M,A),element(A,c)	3/5	3/5	3/19	4/10
mol2atom(M,A),element(A,h)	4/5	2/5	5/19	2/10
mol2atom(M,A),element(A,n)	3/5	1/5	3/19	1/10
mol2atom(M,A),element(A,o)	4/5	3/5	8/19	3/10
mol2atom(M,A),type(A,9)	4/5	2/5	5/21	2/12
mol2atom(M,A),type(A,14)	3/5	3/5	3/21	3/12
mol2atom(M,A),type(A,17)	1/5	2/5	1/21	2/12
mol2atom(M,A),type(A,21)	3/5	1/5	3/21	1/12
mol2atom(M,A),type(A,22)	4/5	3/5	6/21	3/12
mol2atom(M,A),type(A,38)	3/5	1/5	3/21	1/12
Tau	N/A	N/A	1/6	1/3

Example: classification

- 1BC
  - positive: 3/5 x 4/5 x (1-3/5) x 4/5 x 4/5 x (1-3/5) x 1/5 x (1-3/5) x 4/5 x (1-3/5) = $1.3\ 10^{-3}$
  - negative: 3/5 x 2/5 x (1-1/5) x 3/5 x 2/5 x (1-3/5) x 2/5 x (1-1/5) x 3/5 x (1-1/5) = $2.8\ 10^{-3}$

- 1BC2
  - P((o,22)|+) = 8/19 x 6/21 = 48/399
  - P((c,17)|+) = 3/19 x 1/21 = 3/399
  - P((h,9)|+) = 5/19 x 5/21 = 25/399
  - positive: 1/6 x 5/6 x 48/399 x 5/6 x 3/399 x $(5/6 \times 25/399)^3 = 1.5\ 10^{-8}$
  - negative: $8.1\ 10^{-9}$

Conclusions

- 1BC2: treats collections of objects as first-class citizens
- Recursive implementation order of magnitude faster than straight propositionalisation
  - now also available for 1BC
- Both approaches available in single system
  - discretisation
  - cross-validation
  - ROC optimisation of decision threshold