# Learning Bayesian networks with local structure

7.11.2001

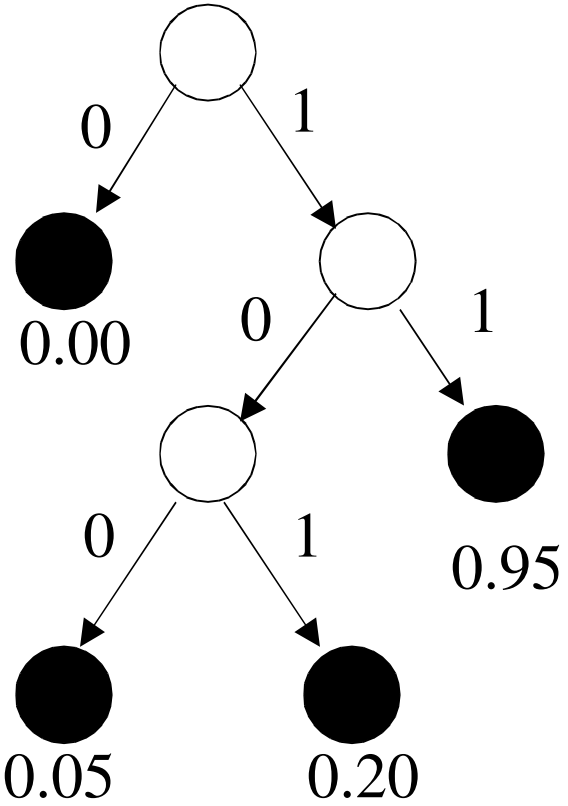
Jarmo Korhonen

# Bayesian Networks

- BNW consists of two components
  - Directed acyclic graph (DAG): the global structure
  - Conditional probability distributions (CPD)
- CPD can be encoded as a tabular representation
- Exponential in size
- Other representations for CPD
  - default table
  - decision tree

# Local CPD

| A | B | E | Pr(S\|A,B,E) |
|---|---|---|---|
| 1 | 1 | 1 | 0.95 |
| 1 | 1 | 0 | 0.95 |
| 1 | 0 | 1 | 0.20 |
| 1 | 0 | 0 | 0.05 |
|  | * |  | 0.00 |



# Learning Bayesian Networks

- Most research to learn global structure and tables
- We can also learn the local structure
  - Context–specific independency (CSI)
  - Context: some assignment to variables in the network (d–separation?)
  - The global structure follows from locals
- Notation:
  - set of variables U, variable X, parents $Pa_i$, parameters ? in CPDs ($?_{xi|pai}$ for each $x_i \in Val(X_i)$ and $pa_i \in Val(Pa_i)$)
  - training set $D = \{u_1,...,u_N\}$, network $B=<G,L>$

# Scoring functions: MDL

- Minimum Description Length
- DL for graph
- DL for CPD of $X_i$
- Total description
- Conditional entropy of given Y: $H(X|Y) = -?_{x,y} P_D(x,y) \log P_D(x|y)$
  - How many bits needed to encode the value of $X_i$, when we know $Pa_i$

$$DL_{graph}(G) = \sum_i \left( \log n + \log \binom{n}{|\mathbf{Pa}_i|} \right)$$

$$DL_{tab}(X_i, \mathbf{Pa}_i) = \frac{1}{2} ||\mathbf{Pa}_i|| (||X_i|| - 1) \log N$$

$$DL(G, D) = DL_{graph}(G) + \sum_i DL_{tab}(X_i, \mathbf{Pa}_i) + N \sum_i H(X_i \mid \mathbf{Pa}_i).$$

$$\hat{P}_D(A) = \frac{1}{N} \sum_{i=1}^{N} 1_A(\mathbf{u}_i) \text{ where } 1_A(\mathbf{u}) = \begin{cases} 1 & \text{if } \mathbf{u} \in A \\ 0 & \text{if } \mathbf{u} \notin A \end{cases}$$

# Scoring functions: BDe

- Equal to posterior probability
- Prob. that network is $G^h$, given the data D
- Maximise Pr
- On large data sets, BDe and MDL converge
- Use Dirichlet priors to get closed form solution

$$\Pr(G^h \mid D) = \alpha \Pr(D \mid G^h) \Pr(G^h)$$

$$\Pr(D \mid G^h) = \int \Pr(D \mid \Theta_G, G^h) \Pr(\Theta_G \mid G^h) d\Theta_G$$

# Local Structure

- Context−specific independency (CSI)
  - $P(X|Z,c,Y)=P(X|Z,c))$
- Partitions: group values of parents together
- Partition variables $Y_L$
  - characteristic random variable
  - maps each value of $Pa_i$ to the partition that contains it
- Partitions in different CPD:s
  - Default tables: default row, other rows
  - Decision trees: paths to leaf nodes

# Learning Local Structure

- Scoring: MDL
  - MDL: $Dl_{local-struct}(D)$ = log ||$Pa_i$|| + log (||$Pa_i$|| over k)
    k = |rows(D)|

$$DL(G, L_1, \ldots, L_n, D) = DL_{graph}(G) + \sum_i (DL_{local\text{-}struct}(L_i) + DL_{param}(L_i)) + N \sum_i H(X \mid \Upsilon_{L_i}).$$

- Scoring: BDe

$$\Pr(D \mid \mathcal{L}_G^h, G^h) = \prod_i \prod_{v \in Val(\Upsilon_{L_i})} \int \prod_{x_i} \theta_{x_i|v}^{N(x_i,v)} \Pr(\Theta_{X_i|v} \mid {L_i}^h, G^h) d\Theta_{X_i|v}$$

# Learning procedures

- Greedy hillclimbing
- Successors by
  - adding an arc
  - removing an arc
  - reversing direction of an arc
- Decomposable
  - need not recalculate everything for scores

procedure LearnNetwork($G_0$)

Let $G_{current}$ •$G_0$

do

generate successors $S=\{G_1-G_n\}$ of $G_0$

?score=$\max_{G\in S}$score(G) − score($G_{current}$)

if ?score > 0 then

let $G_{current}$ arg $\max_{G\in S}$score(G)

while (?score > 0)

return $G_{current}$

# Learning procedures 2

procedure LearnDefault()
Let Rows(D) •Ø
do
let r = arg maxrÎςal(Pa)–rows(D)
    Score(Rows(D)È{r})
if Score(Rows(D)È{r}) <
    Score(Rows(D) then
return Rows(D)
Rows(D) ¥ Rows(D)È{r}
end

```
procedure SimpleTree(Y)
    For y ∈ Val(Y), let l_y ← Λ (i.e., a leaf)
    return ⟨Y, {l_y : y ∈ Val(Y)}⟩
end
procedure LearnTree(D)
    if D = ∅ or X_i is homogeneous in D then
        return Λ.
    // Growing phase
    Let Y_split = arg max_{Y∈Pa_i} Score(SimpleTree(Y) | D)
    for y ∈ Val(Y_split)
        Let D_y = {u_i ∈ D : Y_split = y in u_i}
        Let T_y = ExpandTree( Λ, D_y)
    let T = ⟨Y_split, {T_y : y ∈ Val(Y)}⟩
    // Trimming phase
    if Score(Λ | D) > Score(T | D) then
        return Λ
    else
        return T
end
```

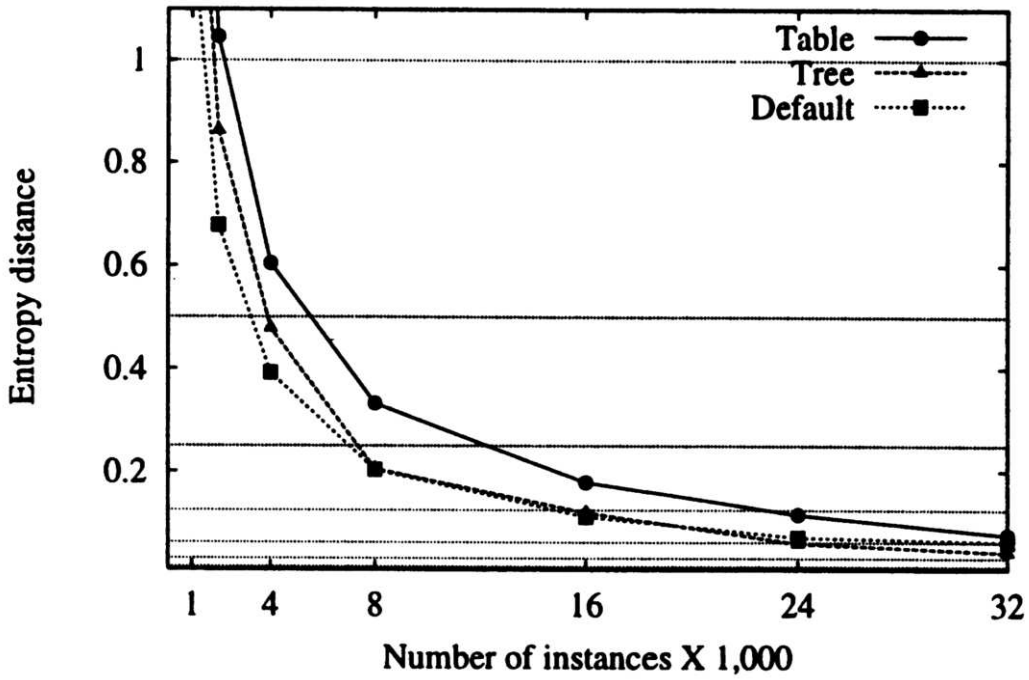
# Experimental Results

- More accurate models, no arbitrary limits
- Fewer parameters needed
- Better approximations
- Comparison between table−based, default tables and decision trees
- Comparison between MDL and BDe scoring
- Variations in algorithm parameters (e.g. priors for BDe)
- Approximately 15,000 networks

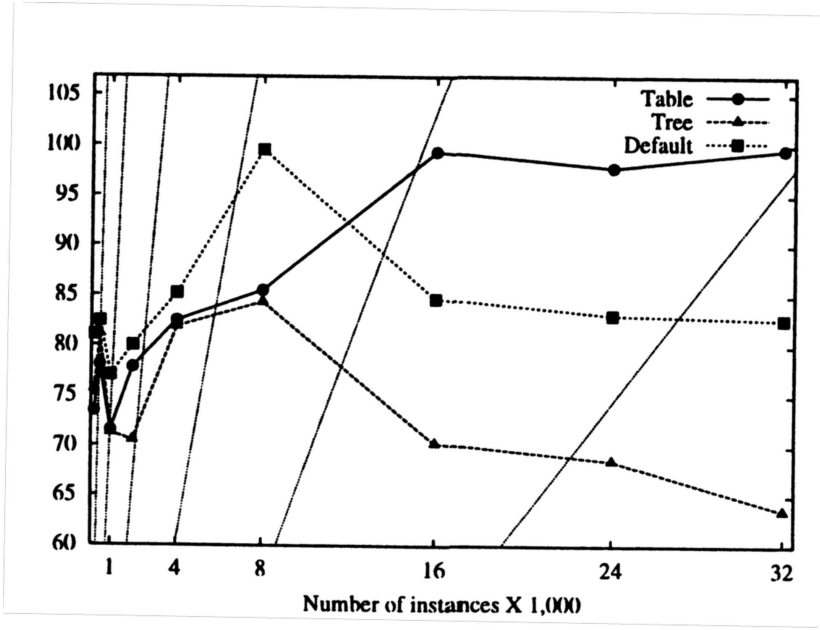
# Testing

- Three known networks
  - Patient alarm in intensive care
  - Hailfinder – model for summer hail
  - Insurance – classify insurance application
- Data generated by known networks
- Error measure: entropy distance
  - $D(P\|Q) = ?_X P(x) \log (P(x)/Q(x))$
  - $D(P\|Q) ? 0$
- Inherent error (from network structure)
  - how sensible the independency assumptions were

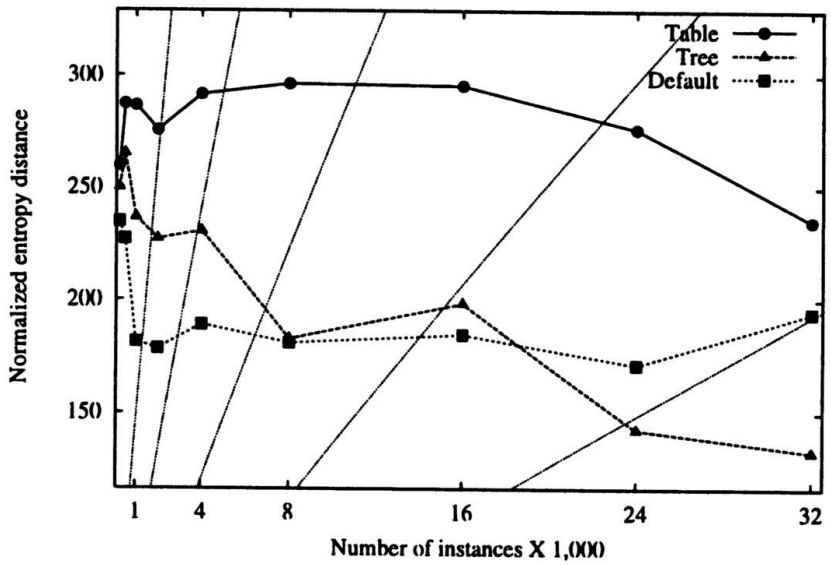
# Results (Alarm, MDL)



# Results (Alarm, normalised)




BDe score
Error curve in
Alarm domain




MDL score
Error curve in
Alarm domain

# Conclusion

- Many ways to represent the local structure
  - here trees and default tables
- Local structure improves performance
- Better models, because less arbitrary limits needed on network structure
- Less data needed
- Improvements:
  - use global structure
  - use better CPDs