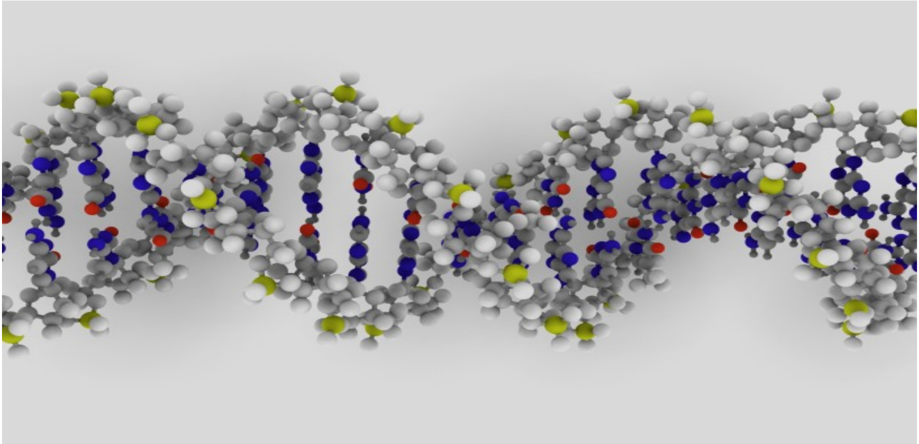
Bayesian Logic Programs

Claus Brech
Moritz Mark

TECHNISCHE UNIVERSITÄT DARMSTADT



Introduction and Motivation

- There has been an increasing interest in integrating probability theory with first order logic within the last years
- Bayesian networks are elegant and efficient probabilistic frameworks, but they inherit disadvantages of propositional logic (see next slides)
- Our presentation introduces Bayesian logic programs
  - They unify Bayesian networks with inductive logic programming and generalize both concepts
  - Main goal: Inherit advantages and overcome limitations of both frameworks
- Key idea: One-to-one mapping between ground atoms (inductive logic programming) and random variables (Bayesian networks)

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Revisiting Bayesian Networks and Logic Programs

2. Bayesian Logic Programs

3. Learning Bayesian Logic Programs

4. Conclusion and Outlook

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Revisiting Bayesian Networks and Logic Programs
1.1 Bayesian Networks
1.2 Logic Programs

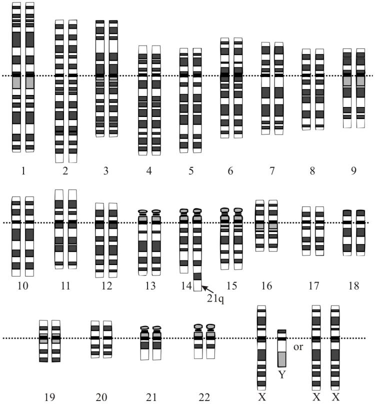
2. Bayesian Logic Programs

3. Learning Bayesian Logic Programs

4. Conclusion and Outlook

Genetics Example

A person's `X` blood type `bt(X)` is determined by a gene that is inherited. Each person `X` has two copies of the chromosome containing this gene. `mc(Y)` inherited from the mother `m(Y,X)` and `pc(Z)` inherited from the father `f(Z,X)`.



The genetic model:

- has a probabilistic part through the biological laws of inheritance, and
- requires the representation of the relational family structure

Bayesian Networks
Basics

- A Bayesian network is a directed acyclic graph
- Each **node** corresponds to a **random variable** x_i
- An **edge** from x_1 to x_2 indicates a **direct influence** from x_1 on x_2

$\Rightarrow$ The Bayesian network represents the joint probability distribution $P(x_1, ..., x_n)$ over the fixed, finite set $\{x_1, ..., x_n\}$ of random variables

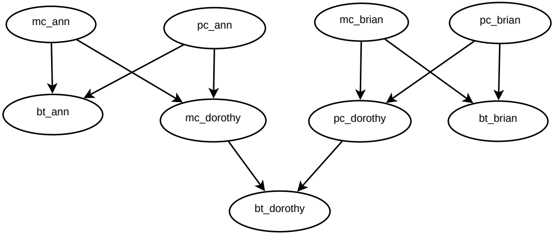
- The random variable x_i possesses the finite set of possible states $S(x_i)$

Possible states:

$S(bt\_dorothy) = \{a, b, ab, 0\}$

$S(pc\_dorothy) = \{a, b, 0\}$

$S(mc\_dorothy) = \{a, b, 0\}$



Bayesian Networks
Joint Probability Density

- The direct predecessors (parents) of a node x are identified by $Pa(x)$
- For instance: $Pa(bt\_ann) = \{pc\_ann, mc\_ann\}$

Independence Assumption of Bayesian Networks: Each node x_i in the graph is conditionally independent of any subset A of nodes that are not descendants of x_i given a joint state of $Pa(x_i)$

$$\Rightarrow P(x_i|A, Pa(x_i)) = P(x_i|Pa(x_i))$$

Applying the independence assumption to the chain rule expression of the joint probability distribution we get the **joint probability density**:

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i|Pa(x_i))$$

And we associate every x_i with a conditional probability distribution

$$P(x_i|Pa(x_i))$$

Bayesian Networks
Conditional Probability Distribution

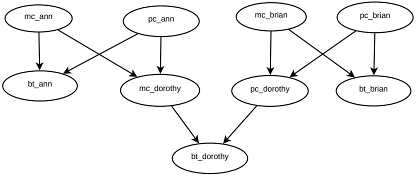
The conditional probability distribution for the blood type example could be:

mc_dorothy	pc_dorothy	P(bt_dorothy)
a	a	(0.97,0.01,0.01,0.01)
b	a	(0.01,0.01,0.97,0.01)
...	...	
0	0	(0.01,0.01,0.01,0.97)

mc_ann	pc_ann	P(mc_dorothy)
a	a	(0.98,0.01,0.01)
b	a	(0.01,0.98,0.01)
...	...	
0	0	(0.01,0.01,0.98)

and for the apriori nodes(nodes having no parents):

P(mc_ann)	P(pc_ann)	P(mc_brian)	P(pc_briany)
(0.38,0.12,0.50)	(0.38,0.12,0.50)	(0.38,0.12,0.50)	(0.38,0.12,0.50)



Bayesian Networks
Pros and Cons

All Bayesian networks consists of two components:

- a *qualitative* or *logical* one, that represents the influences among the random variables using a directed acyclic graph
- a *quantitative* one that encodes the probability densities over these influences, represented by conditional probability tables

Therefore Bayesian networks provide a nice separation of the qualitative and the quantitative component. The **major limitation** of Bayesian networks is their propositional nature. It is not possible to formulate a general probabilistic rule like:

> *the localization L of gene G is influenced by the localization L' of another gene G' that interacts with G'*

Logic Programs
Basics (1)

A Prolog program consists of **clauses**. There are two type of clauses: **facts** and **rules**. Facts describe attributes of an object or a relationship between multiple objects. Facts consist of a functor followed by the list of arguments. A predicate is a functor and its arity. Let's have an example:

```
male(jef).
parent(jef,paul).
parent(paul,ann).
grandparent(X,Y) :-
 parent(X,Z),parent(Z,Y).
```

Existing terms:

- $\underbrace{\textit{male}}_{\textit{functor}} / \underbrace{1}_{\textit{arity}}$ and $\underbrace{\textit{parent}}_{\textit{functor}} / \underbrace{2}_{\textit{arity}}$ are predicates
- `jef`, `paul` and `ann` are *constants*
- `X`, `Y` and `Z` are *variables*

Rules are logical statements like `grandparent(X,Y)`. This rule can be read as: `X` is the grandparent of `Y` if `X` is a parent of `Z` and `T` is a parent of `Y`. Let us call this clause *c*. `grandparent(X,Y)` is called *head*(c) and `parent(X,Z),parent(Z,Y)` is *body*(c) .

Logic Programs
Basics (2)

TECHNISCHE UNIVERSITÄT DARMSTADT

Definitions:

- **Atoms** are predicates followed by the necessary number of terms e.g. `parent(jef,paul)`
- $Var(E)$ are the variables occurring in a term, atom or clause: e.g. $Var(c) = X, Y, Z$
- If $Var(E) = \emptyset$ (no variables occur in term, atom or clause E), E is called *ground*
- A **substitution** $\theta = \{V_1/t_1, ..., V_n/t_n\}$, e.g. $\{X/ann\}$ is an assignment of terms t_i to variables V_i
  $\Rightarrow c\theta$ is `grandparent(ann,Y) :- parent(ann,Z),parent(Z,Y)`

Logic Programs
Least Herbrand Model

- A **Herbrand base** $HB(T)$ of a logic program T is the set of all ground atoms constructed with the symbols in T.
- E.g. $HB(grandparent) = \{$`parent(ann,ann), parent(jef,jef), parent(paul,paul), parent(ann,jef), parent(jef,ann), ..., grandparent(ann,ann), grandparent(jef,jef),...`$\}$
- A **Herbrand interpretation** is a subset of $HB(T)$.
- The **least Herbrand model** $LH(T)$ is the subset of $HB(T)$ containing all ground logical consequences (i.e. all relevant facts)
- There several methods to compute the least Herbrand model
- E.g. $LH(grandparent) = \{$`male(jef), parent(jef,paul), parent(paul,ann), grandparent(jef,ann)`$\}$

Agenda

TECHNISCHE UNIVERSITÄT DARMSTADT

1. Revisiting Bayesian Networks and Logic Programs

2. Bayesian Logic Programs
2.1 Representation Language and Semantics
2.2 Graphical Representation
2.3 Extension to the Basic Framework

3. Learning Bayesian Logic Programs

4. Conclusion and Outlook

Representation Language and Semantics
Previous Findings

Findings of the previous example:

- Random variables from the Bayesian network correspond to logical atoms
- The direct influence relation corresponds to the immediate consequence operator
- The logical component of Bayesian networks correspond to propositional logic (and inherit there limitations)

To overcome the limitations: Bayes logic programs have to upgrade the network structure of the Bayesian networks to first order clauses

$\Rightarrow$ Bayesian Clauses

Representation Language and Semantics
Structural Component

Definition: Bayesian Clauses *A Bayesian clause c is an expression of the form $A|A_1,...,A_n$ where $n \geq 0, A, A_1,...,A_n$ are Bayesian atoms and all Bayesian atoms are universally quantified.*

- A Bayesian predicate p\1 represents a set S(p\1) of random variables
- A Bayesian ground atom g represent random variables over the states S(g)
- '|' is used instead of ':-' to capture the idea of conditional probability distributions
- Range assumption: All Bayesian clauses c are range-restricted i.e. $Var(head(c)) \subseteq Var(body(c))$ to avoid the derivation of non-ground true facts

Example: `bt(X) | mc(X),pc(X)` is a *Bayesian clause*

- E.g. bt(ann) corresponds to a random variable over the states $S(bt(ann)) = \{a, b, ab, 0\}$

Probabilistic model: Each Bayesian clause c is associated with a conditional probability distribution $cpd(c) = P(head(c)|body(c))$

- There may be more than one clause. Considering the following clauses

$$c_1\texttt{: bt(X) | mc(X)}$$
$$c_2\texttt{: bt(X) | pc(X)}$$

- Assume substitutions θ_i that ground the causes c_1 and c_2 so that they fulfill $head(c_1\theta_1) = head(c_2\theta_2)$. θ_i implies $cpd(c_1\theta_1)$ and $cpd(c_2\theta_2)$
- **Problem**: For probabilistic reasoning we need the distribution $P(head(c_1\theta_1)|body(c_1) \cup body(c2))$
- **Solution**: A **Combining Rule** *cr(p/l)* is a function that maps finite sets of probability distributions onto one combined conditional probability distribution

Representation Language and Semantics
Bayesian Logic Programs

TECHNISCHE UNIVERSITÄT DARMSTADT

A **Bayesian logic program** B consists of a set of Bayesian clauses

- For each Bayesian clause c there is exactly one $cpd(c)$
- For each Bayesian predicate p/l there is exactly one $cr(p/l)$

Example: Bayesian logic program for the blood-type domain

```
m(ann, dorothy).
f(brian, dorothy).
pc(ann).
pc(brian).
mc(ann).
mc(brian).

mc(X) | m(Y,X),mc(Y),pc(Y).
pc(X) | f(Y,X),mc(Y),pc(Y).
bt(X) | mc(X),pc(X).
```

mc_dorothy	pc_dorothy	P(bt_dorothy)
a	a	(0.97,0.01,0.01,0.01)
b	a	(0.01,0.01,0.97,0.01)
...	...	
0	0	(0.01,0.01,0.01,0.97)

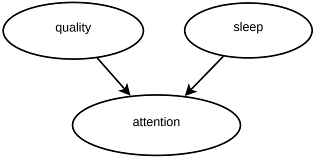
m(ann,dorothy)	mc_ann	pc_ann	P(mc_dorothy)
true	a	a	(0.98,0.01,0.01)
true	b	a	(0.01,0.98,0.01)
...	...		
false	0	0	(0.33,0.33,0.33)

Representation Language and Semantics
Declarative Semantics (1)

A Bayesian logic program B can be visualized by a **dependency graph** $DG(B)$

- The **nodes** are the atoms in the least Herbrand model of the Bayesian logic program
- There is an **edge** between a node x and a node y if there exists a clause $c \in B$ and a substitution θ, s.t. $y = head(c\theta)$, $x \in body(c\theta)$ and for all ground atoms z in $c\theta$: $z \in LH(B)$
- Application of the combining rule $cr(p/n)$ allows us to assign a **conditional probability distribution** to the nodes having parents

```
quality.      attention | quality
sleep.        attention | sleep
```



Representation Language and Semantics
Declarative Semantics (2)

TECHNISCHE UNIVERSITÄT DARMSTADT

A Bayesian logic program B that satisfies

1. $LH(B) \neq \emptyset$
2. $DG(B)$ is acyclic
3. each node in $DG(B)$ is influenced by a finite set of random variables

is called **well-defined**.

- Every *well-defined* Bayesian logic program specifies a unique joint distribution over LH(B)
- The joint distribution can be factored to

$$P(LH(B)) = \prod_{x \in LH(B)} P(x | Pa(x))$$

Representation Language and Semantics
Query-Answering Procedure

A **probabilistic query** to a Bayesian logic program *B* can be defined as an expression of the form:

$$?-q_1, ..., q_n | e_1 = \mathrm{e}_1, ..., \mathrm{e}_m = e_m$$

where $n > 0$, $m \geq 0$. It asks for the conditional probability distribution

$$P(q_1, ..., q_n | e_1 = e_1, ..., e_m = e_m)$$

of the query variables $q_1, ..., q_n$ where $\{q_1, ..., q_n, e_1, ..., e_n\} \subseteq HB(B)$.

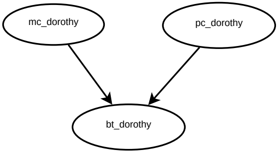
One naive approach would be to compute the whole least Herbrand model to answer queries

In order to answer queries, one has not compute the whole least Herbrand model. One way to avoid complex network structures is to consider so-called **support networks**.

$$N = \{x\} \cup \{y | y \in LH(B) \text{ and y influences } x\}$$

The **support network** N of a random variable $x \in LH(B)$ consists of x and all variables y that influence x

Example: Support Network for `bt(dorothy)`

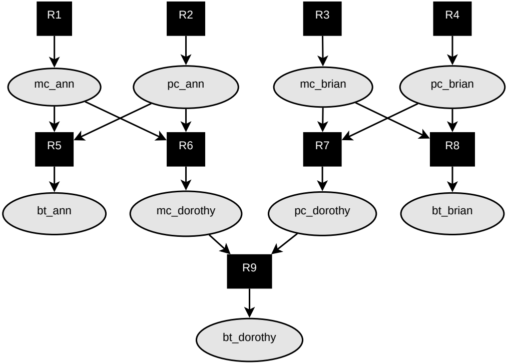


Graphical Representation
Extended Bayesian Network Representation (1)

TECHNISCHE UNIVERSITÄT DARMSTADT

Extended representation for Bayesian networks:

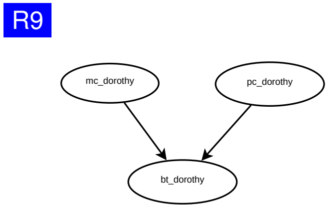
- **Bipartite** directed acyclic graph
- Two disjoint sets of nodes in the bipartite graph:
  1. *light gray ovals* are random variables
  2. *black boxes* are local probability models
- The local probability models specify the conditional probability distribution $P(\mathtt{x_i} | Pa(\mathtt{x_i}))$



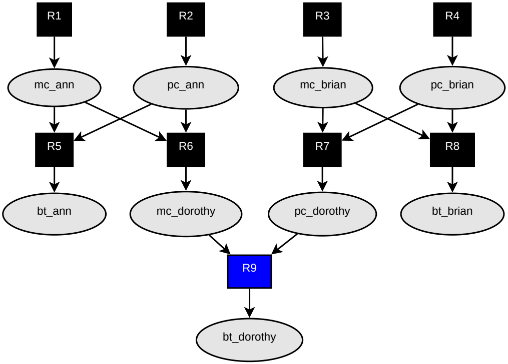
Graphical Representation
Extended Bayesian Network Representation (2)

TECHNISCHE UNIVERSITÄT DARMSTADT

The local probability model for R9

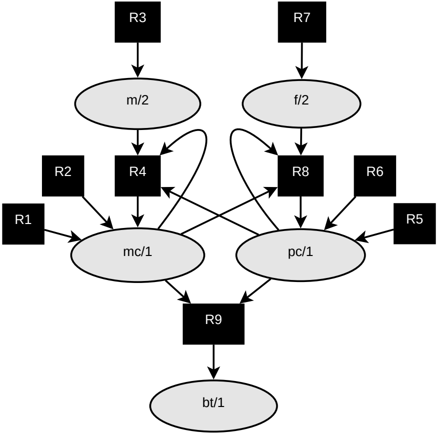


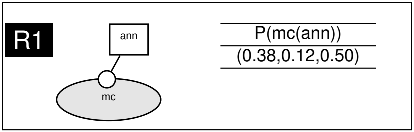
mc_dorothy	pc_dorothy	P(bt_dorothy)
a	a	(0.97,0.01,0.01,0.01)
b	a	(0.01,0.01,0.97,0.01)
...	...	
0	0	(0.01,0.01,0.01,0.97)



R9 specifies the conditional probability distribution for $P(\mathtt{bt}(\mathtt{dorothy})|\mathtt{mc}(\mathtt{dorothy}),\mathtt{pc}(\mathtt{dorothy}))$

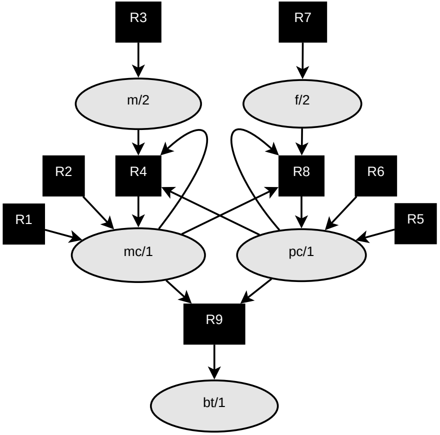
Graphical Representation for Bayesian Logic Programs (1)

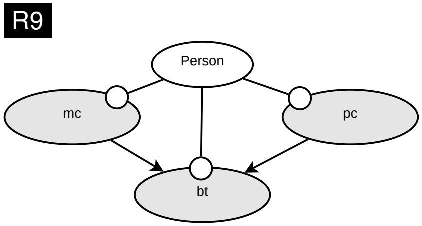




- Constants and functors such as `ann` are represented as white boxes
- Bayesian atoms are represented as gray ovals containing the predicate: `pc`
- Arguments are represented as white empty circles on the boundary of the ovals

Graphical Representation for Bayesian Logic Programs (2)

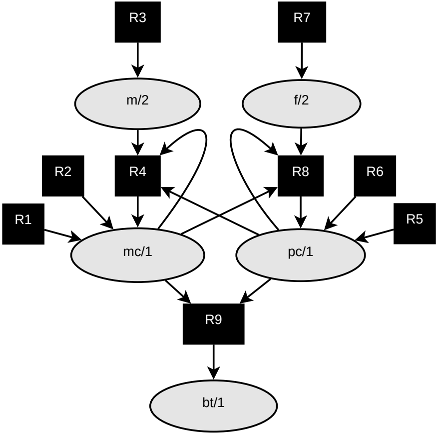


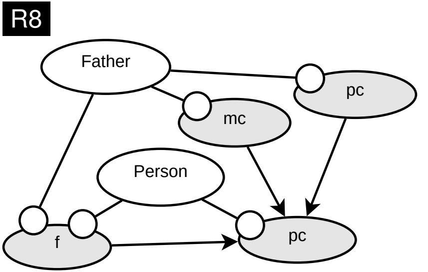


mc(Person)	pc(Person)	P(bt(Person))
a	a	(0.97,0.01,0.01,0.01)
b	a	(0.01,0.01,0.97,0.01)
...	...	...
0	0	(0.01,0.01,0.01,0.97)

- Arguments of atoms like `Person` are placeholders for terms.

Graphical Representation for Bayesian Logic Programs (3)

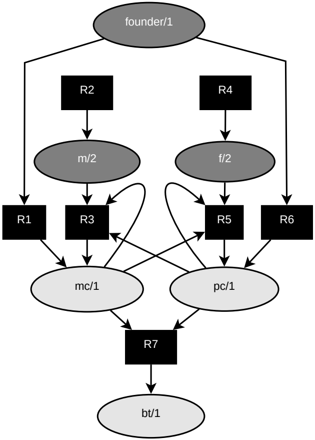




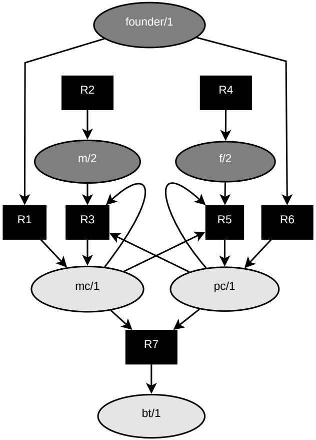
f(Father,Person)	mc(Father)	pc(Father)	P(pc(Person))
true	a	a	(0.98,0.01,0.01)
true	b	a	(0.01,0.98,0.01)
...	...	...	...
false	0	0	(0.33,0.33,0.33)
...	...	...	...

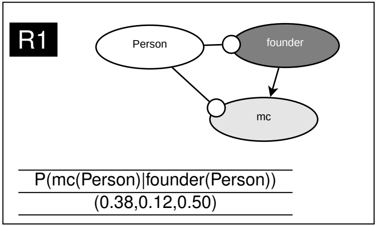
Graphical Representation for Bayesian Logic Programs
Logical Atoms (1)

- The `mother/2` and `father/2` relations are not really *random* variables, they are always in the same state. (*true* with probability 1)
  $\Rightarrow$ `mother/2` and `father/2` are *logical* atoms
- Logical atoms have no conditional probability distribution
- `pc(X)|f(Y,X),mc(Y),pc(Y)` is modified to `pc(X)|mc(Y),pc(Y)` and it only applies for substitutions for which `f(Y,X)` is true (in the least Herbrand model)
- *Bayesian* atoms are light gray ovals, *logical* atoms are dark gray ovals



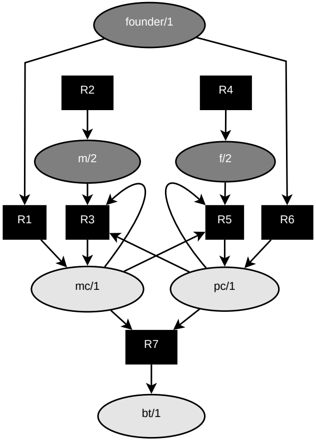
Graphical Representation for Bayesian Logic Programs
Logical Atoms (2)

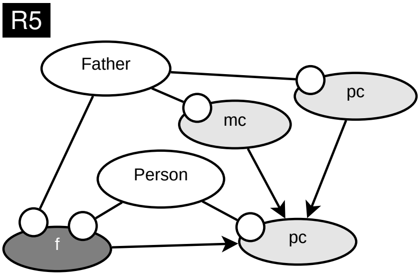




- The new `foundation/1` relation is defined as:
```
founder(Person):-
    \+(mother(_,Person);father(_,Person))
```
- `\+` is a negation, `_` represents a anonymous variable and `;` stands for disjunction
- Essentially `founder(Person)` is true if `Person` has no parents (is an apriori node)

Graphical Representation for Bayesian Logic Programs
Logical Atoms (3)

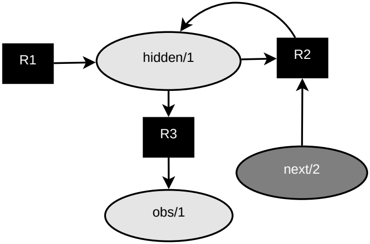


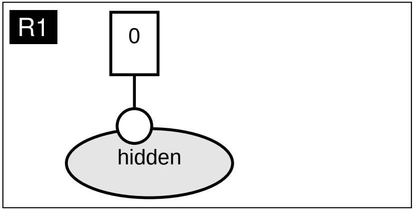


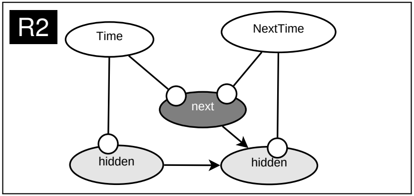
mc(Father)	pc(Father)	P(pc(Person))
a	a	(0.98,0.01,0.01)
b	a	(0.01,0.98,0.01)
...	...	...
0	0	(0.33,0.33,0.33)
...	...	...

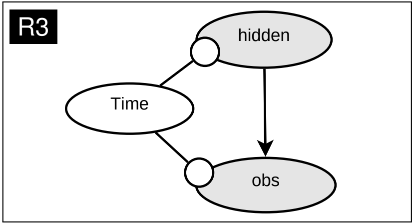
Graphical Representation for Bayesian Logic Programs
Another example

TECHNISCHE UNIVERSITÄT DARMSTADT

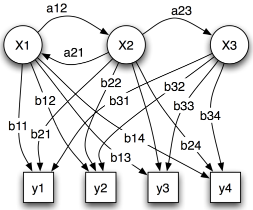








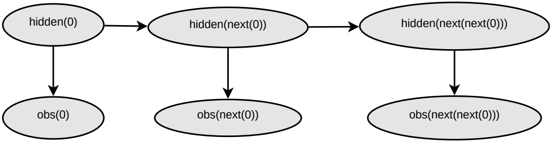
Graphical Representation for Bayesian Logic Programs
Hidden Markov Models (1)



left General hidden Markov model
Parameters:

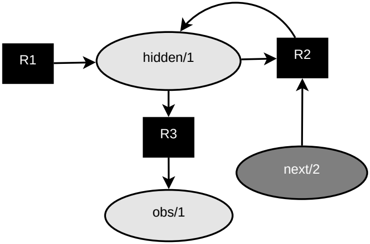
- **x** states
- **y** possible observations
- **a** state transition probabilities
- **b** output probabilities

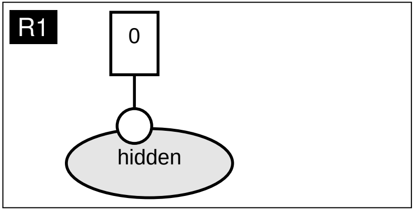
bottom The graph of the Bayesian logic program on the previous slide directly encodes the Bayesian network structure of hidden Markov models

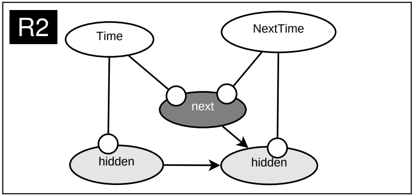


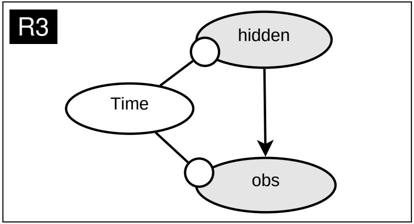
Graphical Representation for Bayesian Logic Programs
Hidden Markov Models (2)

TECHNISCHE UNIVERSITÄT DARMSTADT









Agenda

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Learning Bayesian Logic Programs

TECHNISCHE UNIVERSITÄT DARMSTADT

Previously:

- Assumption: There is an expert who designs the structure and also the conditional probability of the network
- Problem: Lack of persons with the necessary expertise or knowledge
- Solution: If there is access to data, there is a possibility learn Bayesian logic programs

Learning approach:

- Learning based on given data cases (*learning from interpretations*)
- Key idea: Rules that are valid on one interpretation are likely to be valid on other interpretations

The Learning Setting
Data Cases

A **data case** $D_i \in \mathbf{D}$ consists of a

- Logical part: $Var(D_i) = LH(B \cup Var(D_i))$ (Herbrand interpretation) and a
- Probabilistic part: Assignment of values to some facts in $Var(D_i)$

Example of a data case:

$D_1 = \{m(cecily, fred) = true, f(henry, fred) =?, pc(cecily) = a, pc(henry) = b, pc(fred) =?, mc(cecily) = b, mc(henry) = b, mc(fred) =?, bt(cecily) = ab, bt(henry) = b, bt(fred) =?\}$

- The logical part specifies the least Herbrand model of the target Bayesian program and highlights *relevant* random variables
- The probabilistic part induce a joint distribution over the random variables of the logical part

The Learning Setting
Hypothesis Space

The **hypothesis space** consists of Bayesian logic programs, i.e.

- A finite set of Bayesian clauses
- Associated conditional probability distributions

Adequate **restrictions** have to be designed to define the hypothesis space, e.g.

- The clauses only contain constant and predicate symbols that occur in one of the data cases
- One clause is limited to only 3 atoms

Not every element of the hypothesis space is an adequate candidate. Possible candidates have to fulfill the following requirements:

- It has to be logically *valid* on the data
- The induced Bayesian network has to be *acyclic*

The Learning Setting
Formulation of the Learning Problem

TECHNISCHE UNIVERSITÄT DARMSTADT

Given:

- A set $D = \{D_1, ..., D_m\}$ of data cases
- A set H of hypothesis
- Scoring function $score_D : H \mapsto \mathbb{R}$

Find: A hypothesis $H^* \in \mathbb{H}$ such that

- For all $D_i \in \mathbb{D} : Var(D_i) = LH(H^* \cup Var(D_i))$
- The Bayesian network implied by H^* is acyclic
- H^* maximizes $score_D : H \mapsto \mathbb{R}$

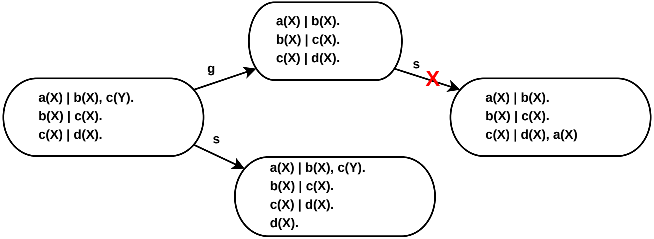
Assumption: An adequate *score function* is given that expresses how well a given candidate $H \in \mathbb{H}$ fits the given Data. Example for a scoring function: Likelihood.

Structural Learning of Bayesian Logic Programs
Scooby (1)

Scooby: **S**tru**c**tural learning **o**f intensi**o**nall **B**a**y**esian logic programs

- performs a *heuristic search* through the hypothesis space
- uses the *refinement operators* ρ_g and ρ_s to traverse the hypotheses space
- Works for the special case of Bayesian networks only

Example of the use of the refinement operators:



Structural Learning of Bayesian Logic Programs Scooby (2)

Scooby: <u>**S**</u>tru<u>**c**</u>tural learning <u>**o**</u>f intensi<u>**o**</u>nall <u>**B**</u>a<u>**y**</u>esian logic programs

Let H be an initial (valid) hypothesis; $S(H) := score_D(H)$;
ρ_g and ρ_s are generalization and specialization operators
repeat
    $H' := H$; $S(H') := S(H)$;
    for all $H'' \in \rho_g(H') \cup \rho_s(H')$ **do**
        if H'' is (logically) valid on D **then**
            if the Bayesian networks induced by H'' on the data are acyclic **then**
                if $score_D(H'') > S(H)$ **then**
                    $H := H''$
                    $S(H) := S(H'')$
                end if
            end if
        end if
    end for
until $S(H') = S(H)$
return H;

- ρ_g deletes a Bayesian proposition from the body of a clause
- ρ_s adds a Bayesian proposition to the body of a clause

Structural Learning of Bayesian Logic Programs
Scooby (3)

In order to **adapt Scooby** to work in the case of Bayesian networks one has to consider:

1. Some Bayesian logic programs will be logically invalid
   - Initialization of valid Bayesian logic programs
   - Filtering out those Bayesian logic programs that are logically invalid
2. The traditional first order refinement operator must be used
   - Instead of adding/deleting propositions, they add/delete constant-free atoms

Ideas to improve the algorithms performance:

- **Lookahead**: Allowing atoms to be chosen that do not result in a better score to avoid local optima of the score function
- Include **Background Knowledge** i.e. fixed regularities that are common to all examples
- **Improve Scoring Function** e.g. use of *minimal description length* principle

Learning Probabilities

Previously we assumed that there is a given method to calculate the conditional probability distribution. Parameter Estimation:

- **Given:** A set $D = \{D_1, ..., D_n\}$ and a set H of Bayesian clauses, a scoring function $score_D : \mathbb{H} \mapsto \mathbb{R}$
- **Find:** Parameters of H maximizing $score_D$

Maximum Likelihood is used as the method for parameter estimation. As parameter estimation techniques

- Gradient-based approaches and
- Expectation-Maximization algorithm

will be discussed.

Learning Probabilities
Maximum Likelihood Estimation

Given: Bayesian logic program B consisting of Bayesian clauses $c_1, ..., c_n$, data cases $D = \{D_1, ..., D_n\}$

The **likelihood** $L(D|\lambda)$ is the probability of the data D as a function of unknown parameters λ

$$L(D|\lambda) := P_B(D|\lambda)$$

Task: Find parameter values λ^* that maximize the likelihood

$$\lambda^* = \max_{\lambda \in H} P_B(D|\lambda) = P_{B(\lambda)}(D)$$

Using the fact that it is sufficient to consider the support network N. Due to the monotonicity of the logarithm, we can formulate the problem like that:

$$\lambda^* = \max_{\lambda \in H} log\, P_{N(\lambda)}(D)$$

Learning Probabilities
Gradient-Based approach

TECHNISCHE UNIVERSITÄT DARMSTADT

According to the **Gradient-Based approach** the chain rule is applied and parameters λ are fixed

$$\frac{\delta log\, P_N(D)}{\delta cpd(c_i)_{jk}} = \sum_{subst.\theta s.t.sn(c_i\theta)} \frac{\delta log\, P_N(D)}{\delta cpd(c_i\theta)_{jk}} \qquad (1)$$

with grounding substitutions θ and $sn(c_i\theta)$ is true iff $\{head(c_i\theta)_{jk}\} \cup body(c_i\theta) \subset N$ Equation (1) can be rearranged to:

$$\frac{\delta log\, P_N(D)}{\delta cpd(c_i)_{jk}} = \frac{en(c_{ijk}|\theta, D)}{cpd(c_i\theta)_{jk}} \qquad (2)$$

where $en(c_{ijk}|\theta, D) := \sum_{l=1}^{m} P_N(head(c_i\theta) = u_j, body(c_i\theta) = \mathbf{u_k}|D_l)$ are the **expected counts.**

Learning Probabilities
Gradient-Based approach (Algorithm)

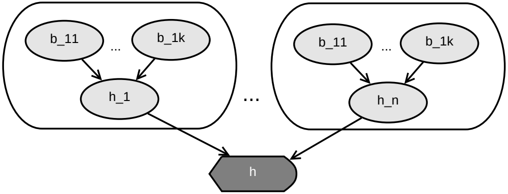
TECHNISCHE UNIVERSITÄT DARMSTADT

```
input:Bayesian logic program B, associated cpds parameterized by λ, data cases D
output: Modified Bayesian logic program B
λ ←INITIAL PARAMETERS
N ←SUPPORTNETWORK(B,D)
repeat
    Δλ ← 0
    set associated conditional probability distribution of N according to λ
    for all D_l ∈ D do
        set evidence in N from D_l
        for all Bayesian clause c ∈ B do
            for all ground instance cθ s.t. {head(cθ)} ∪ body(cθ) ⊂ N do
                for all single parameter cpd(cθ)_jk do
                    Δcpd(c)_jk ← cpd(c)_jk + (δlogP_N(D_l)/δcpd(c)_jk)
                end for
            end for
        end for
    end for
    Δλ ← PROJECTIONONTOCONSTRAINTSURFACE(Δλ)
    λ ← λ + α ∗ Δλ
until Δλ ≈ 0
Return B
```

Learning Probabilities
Gradient-Based approach

The algorithm is relatively simplified. There are **two important points** left out to keep it simple:

1. **Constraint satisfaction**
   - λ consists of probability values $\Rightarrow \sum_j cpd(c_i)_{jk} = 1$
   - Way to enforce this: Re-parameterizing the problem i.e. design of parameters that automatically respect the constraint
2. **Decomposable combining rules** are assumed



- Each group corresponds to a ground instance
- The state of h is a deterministic function of the parents joint state

TECHNISCHE UNIVERSITÄT DARMSTADT

Expectation-Maximization algorithm:

- Another approach to estimate parameters of the maximum likelihood function
- Used in the presence of missing values

If *all values are available* then the parameters corresponds to **frequency counting**:

$$P(X = x|Pa(X) = \mathbf{u}) = \frac{n(X = x, Pa(X) = \mathbf{u_k}|D_l)}{n(Pa(X) = \mathbf{u_k}|D_l)}$$

where $n(\mathbf{a}|\mathbf{D})$ denotes the count of a state **a** given data D.
If some *values are missing* the EM-algorithm performs two steps:

1. **E-Step:** Calculation of a distribution over all possible completions[1] given λ and D
2. **M-Step:** Computes the parameters that maximize the log likelihood function

[1] There is one completion for each partially observed data case. The completion are treated as weighted fully-observed data cases

Learning Probabilities
Gradient vs. EM

Both approaches:

- Rely on computing expected counts
- Perform a greedy local search

Key differences are:

- EM is easier to implement. Reason: EM does not have to enforce the constraint that the parameters are probabilities
- The EM converges faster to near by optimal solutions
- Gradient approaches are more flexible than EM as they allow to consider other scoring functions

Idea: Use of EM and then switch to gradient, since EM converges slowly when it is at a near by optimal solution

Test Cases and Experiments
Genetic Domain

TECHNISCHE UNIVERSITÄT DARMSTADT

Setting:

- Goal was to learn the *bloodtype* program that was used as example
- Two families with 12 respectively 15 members
- For each family 1000 data cases were given (total 2000)
- 40% of the given data was missing

Results:

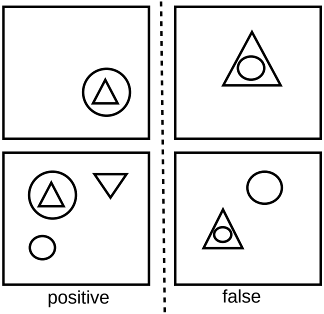
- Fixing the definitions for `m/2` and `f/2` the hypotheses that scored best included: `bt(X) | mc(X), pc(X)`

⇒ The logical structure to generate the data cases was re-discovered

- The definitions of `mc/1` and `pc/1` considered the information from the grandparent to be important, the predicates `m/2` and `f/2` were not used

Test Cases and Experiments
Bongard Domain (1)

TECHNISCHE UNIVERSITÄT DARMSTADT



Setting 1:

- 20 positive and 20 negative examples of the concept *'there is a triangle in a circle'*
- $\{class(e_1) = pos, obj(e_1, o_1) = triangle, obj(e_1, o_2) = circle, class(e_2) = false, obj(e_2, o_1) = triangle, ...\}$
- Given background knowledge $in(e_1, o_1, o_2)$
- 20% missing values in the dataset

Results:

- With $obj(Ex, o_2)$ as lookahead for $in(Ex, o_1, o_2)$ the best scored hypothesis is: $class(Ex)|obj(Ex, o_1), in(Ex, o_1, o_2), obj(Ex, o_2)$
- Without the lookahead the correct hypothesis was not considered

Test Cases and Experiments Bongard Domain (2)

Setting 2 was based on the structure of the program learned in setting 1

- Goal: Estimating the parameters
- 20 positive and 20 negative examples of the disjunctive concept *'there is a (triangle or a circle) in a circle'*

⇒ The probability for the object o_1 to be triangle or a circle was equal (no significant difference)

In **Setting 3** background knowledge about $in(e_1, o_1, o_2)$ was not given

⇒ The algorithm did not discover the correct rule

Setting 4 was a clustering experiment

- 20 positive and 20 negative examples of the concept *'there is a triangle'*
- The examples had 2 to 8 triangles, circles or squares
- 20% of the class labels were missing

⇒ The algorithm learned class(X) | obj(X,Y) which separated the classes perfectly

Test Cases and Experiments
KDD Cup 2001

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Task**: Predict the localization of a given gene in a cell among 15 distinct positions
- **Data**: Relation table with six categorical attributes *Essential, Class, Complex, Phenotype, Motif, Chromosome Number*
- Training set has 862 genes and the test set 381 genes

$\Rightarrow$ The naive Prolog representation (4.400 random variables with over 60.000 parameters) broke cause of memory limits in Sicstus Prolog

- Due to this limitation, the logical structure was based on naive Bayes and only the parameters were estimated
- Estimating the parameters took 12 iterations (about 30 min.)

$\Rightarrow$ The learned Bayesian logic Program achieved an accuracy of 0.57 (top 50% of the submitted models was 0.61, best submission was 0.72)

- The correct localization of a gene was among the top 3 classifications in 77% of the test cases

Agenda

1. Revisiting Bayesian Networks and Logic Programs
2. Bayesian Logic Programs
3. Learning Bayesian Logic Programs
4. Conclusion and Outlook

Conclusion and Outlook (1)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Bayesian logic programs link up Bayesian networks with inductive logic programming
- Any type of Bayesian network and all types of 'pure' Prolog programs can be both represented with Bayesian logic networks. Possible implementations for the blood type example and hidden Markov models were introduced
- They combine the advantages of both Bayesian networks and definite clause logic, including the nice separation into
  - a qualitative part: logical structure of the domain
  - and a quantitative part: conditional probability distributions between the objects of the domain
- Along with strict separation comes the graphical representation

Conclusion and Outlook (2)

- A framework for learning Bayesian network was introduced
  - SCOOBY is used to learn the structure of Bayesian logic program
  - To learn the probabilities Maximum Likelihood approach was used. To estimate the parameters for the Maximum Likelihood two techniques were considered
    - Gradient based approach
    - Expectation Maximization algorithm
- In the experiments the principle of the algorithm is shown and some of the test cases brought good results
- For a general use, the framework has to be improved. This could be done with:
  - An enhanced scoring function for SCOOBY
  - Intensified research on the refinement operator to add or delete atoms
  - Also data cases that do not represent the complete logical structure should be considered respectively

References

TECHNISCHE UNIVERSITÄT DARMSTADT

- Bishop, C. M.: Pattern Recognition and Machine Learning. Springer Verlag, 2007.
- Kersting, K.; De Raedt, L.: Bayesian Logic Programming: Theory and Tool. In Getoor, L.; Taskar, B.: Introduction to Statistical Relational Learning. MIT Press, 2007.
- Kersting, K.; De Raedt, L.: Bayesian Logic Programming: Theory and Tool. In De Raedt, L., Frasconi, P.; Kersting, K.; Muggleton S. (eds.): Probabilistic Inductive Logic Programming. Springer Verlag, 2009.
- Kersting, K.; De Raedt, L.: Bayesian Logic Programs. http://arxiv.org/abs/cs/0111058v1 2000.
- Muggleton, S.: Inductive logic programming. Ohmsha, Ltd., 1990.
- Rabiner, L. R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, Vol. 77, No. 2, 1989
- Roehner, G.: Informatik mit Prolog 1. HIBS, 1995