Bayesian Networks Learning

Fabrizio Riguzzi
Engineering Department

university of ferrara
600 YEARS OF LOOKING FORWARD.

Summary

- Probability theory
- Conditional independence
- Definition of Bayesian network
- Inference
- Learning
- Logic and probability

Uncertainty

- Reasoning requires simplifications:
  - Birds fly
  - Smoke suggests fire
- Treatment of exceptions
- How to reason from uncertain knowledge?

How to Perform Inference?

- Use non-numerical techniques
  - Logicist: non monotonic logic
- Assign to each proposition a numerical measure of uncertainty
  - Neo-probabilist: use probability theory
  - Neo-calculist: use other theories:
    - fuzzy logic
    - certainty factors
    - Dempster-Shafer

Probability Theory

- A: Proposition,
  - Ex: A=The coin will land heads
- P(A): probability of A
- Frequentist approach: probability as relative frequency
  - Repeated random experiments
  - P(A) is the fraction of experiments in which A is true
- Bayesian approach: probability as a degree of belief
- Example: B=burglary tonight

Axioms of Probability Theory

$$0 \leq P(A) \leq 1$$

$$P(\text{Sure Proposition}) = 1$$

$$P(A \vee B) = P(A) + P(B)$$
if A and B are mutually exclusive

Probability Rules

- Any event A can be written as the or of two disjoint events (A and B) and (A and ¬B)

$$P(A) = P(A, B) + P(A, \neg B)$$ **marginalization/ sum rule**

- Where P(A,B)=P(A∧B) is called the **joint probability** of A and B
- In general, if B_i i=1,2,...,n is a set of exhaustive and mutually exclusive propositions

$$P(A) = \sum_i P(A, B_i)$$

- Moreover $$P(A) + P(\neg A) = 1$$

Conditional Probabilities

- P(A|B)= belief of A given that I know B
- Relation to P(A,B)

$$P(A, B) = P(A|B) P(B)$$ **product rule**

$$P(A|B) = \frac{P(A, B)}{P(B)}$$

Bayes Theorem

- Relationship between P(A|B) and P(B|A)

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

- P(A): **prior probability**
- P(A|B): **posterior probability** (after learning B)

Conditional Independence

- If P(A|B)=P(A) we say that A and B are independent
- If P(A|B,C)=P(A|C) we say that A and B are conditionally independent given C

Chain Rule

- n events $E_1,...,E_n$
- Joint event $(E_1,...,E_n)$

$$P(E_{1,}\dots,E_n)=P(E_n|E_{n-1}\dots,E_1)P(E_{n-1},\dots,E_1)$$
$$P(E_{1,}\dots,E_{n-1})=P(E_{n-1}|E_{n-2}\dots,E_1)P(E_{n-2},\dots,E_1)$$
$$\dots$$

- Chain rule:

$$P(E_{1,}\dots,E_n)=P(E_n|E_{n-1}\dots,E_1)\dots P(E_2|E_1)P(E_1)= \prod_{i=1}^{n} P(E_i|E_{i-1},\dots E_1)$$

Multivalued Hypothesis

- Propositions can be seen as binary variables, i.e. variables taking values true or false
  - Burglary B: true or false
- Generalization: multivalued variables
  - Semaphore S, values: green, yellow, red
  - Propositions are a special case with two values

Discrete Random Variables

- Variable V, values v_i i=1,...,n
- V is also called a **discrete random variable**
- $V=v_i$ is a proposition
- Propositions $V=v_i$ i=1,...,n exhaustive and mutually exclusive
- $P(v_i)$ stands for $P(V=v_i)$
- V is described by the set $\{P(v_i)|i=1,...,n\}$, the **probability distribution** of V, indicated with P(V)

Notation

- We indicate with v a generic value of V
- Set or vector of variables: **V**,values **v**

Marginalization

- Multivalued variables A and B
- b_i i=1,...,n values of B

$$P(a)=\sum_i P(a,b_i)$$

- Or

$$P(a)=\sum_b P(a,b)$$

- In general

$$P(\boldsymbol{x})=\sum_{\boldsymbol{y}} P(\boldsymbol{x},\boldsymbol{y}) \qquad \textbf{sum rule}$$

Conditional Probabilities

- P(a|b)= belief of A=a given that know B=b
- Relation to P(a,b)

$$P(a,b)=P(a|b)P(b) \qquad \textbf{product rule}$$

$$P(a|b)=\frac{P(a,b)}{P(b)}$$

- Bayes theorem

$$P(a|b)=\frac{P(b|a)\,p(a)}{P(b)}$$

Continuous Random Variables

- A multivalued variable V that takes values from a real interval [a,b] is called a **continuous random variable**
- P(V=v)=0, we want to compute P(c≤V≤d)
- V is described by a **probability density function** ρ: [a,b]→[0,1]
- ρ(v) is such that

$$P(c \leq V \leq d) = \int_c^d \rho(v)\,dv$$

Properties of Continuous Random Variables

- The same as those of discrete random variables where summation is replaced by integration:
- Marginalization (sum rule)

$$\rho(\boldsymbol{x}) = \int \rho(\boldsymbol{x}, \boldsymbol{y})\,d\,\boldsymbol{y}$$

- Conditional probability (product rule)

$$\rho(\boldsymbol{x}, \boldsymbol{y}) = \rho(\boldsymbol{x}|\boldsymbol{y})\,\rho(\boldsymbol{y})$$

....

Mixed Distribution

- We can have a conjunction of discrete and continuous variables
- Joint: if one of the variables is continuous, the joint is a density:
  - X discrete, Y continuous: ρ(x,y)
- Conditional joint:
  - X discrete, Y continuous: P(x|y)
  - X discrete, Y continuous, Z discrete: ρ(x,y|z)

Domain Modeling

- We use a set of random variables to describe the domain of interest
- Example: home intrusion detection system, variables:
  - Earthquake E, values e_1=no, e_2=moderate, e_3=severe
  - Burglary B, values: b_1=no, b_2=yes through door, b_3=yes through window
  - Alarm A, values a_1=no, a_2=yes
  - Neighbor call N, values n_1=no, n_2=yes

Inference

- We would like to answer the following questions
  - What is the probability of a burglary through the door? (compute $P(b_2)$, belief computation)
  - What is the probability of a burglary through the window given that the neighbor called ? (compute $P(b_2|n_2)$, belief updating)

Inference

- What is the probability of a burglary through the door given that there was a moderate earthquake and the neighbor called ? (compute $P(b_2|n_2,e_2)$, belief updating)
- What is the probability of a burglary through the door and of the alarm ringing given that there was a moderate earthquake and the neighbor called ? (compute $P(a_2,b_2|n_2,e_2)$, belief updating)
- What is the most likely value for burglary given that the neighbor called ($\text{argmax}_b\ P(b|n_2)$, belief revision)

Types of Problems

- Diagnosis: P(cause|symptom)=?
- Prediction: P(symptom|cause)=?
- Classification: argmax_{class} P(class|data)

Inference

- In general, we want to compute the probability $P(\mathbf{q}|\mathbf{e})$
  - of a query $\mathbf{q}$ (assignment of values to a set of variables $\mathbf{Q}$)
  - given the evidence $\mathbf{e}$ (assignment of values to a set of variables $\mathbf{E}$)

Joint Probability Distribution

- The **joint probability distribution** (jpd) of a set of variables **U** is given by P(**u**) for all values **u**
- For our example
  - **U**={E,B,A,N}
  - We have the jpd if we know P(**u**)=P(e,b,a,n) for all the possible values e, b, a, n.

Inference

- If we know the jpd, we can answer all the possible queries:

$$P(\boldsymbol{q}|\boldsymbol{e})=\frac{P(\boldsymbol{q},\boldsymbol{e})}{P(\boldsymbol{e})}$$
$$=\frac{\sum_{x,X\in \boldsymbol{U}\setminus \boldsymbol{Q}\setminus \boldsymbol{E}} P(x,\boldsymbol{q},\boldsymbol{e})}{\sum_{x,\boldsymbol{X}\in \boldsymbol{U}\setminus \boldsymbol{E}} P(x,\boldsymbol{e})}$$

Problem

- If we have n binary variables (|**U**|=n), knowing the jpd requires storing $O(2^n)$ different values.
- Even if had the space to store all the 2^n different values, computing P(**q**|**e**) would require $O(2^n)$ operations
- Impractical for real world domains
- How to avoid the space and time problems? Use conditional independence assertions

Conditional Independence

- **X**, **Y**, **Z** vectors of multivalued variables
- **X** and **Y** are **conditionally independent** given **Z** if

$$P(\boldsymbol{x}|\boldsymbol{y},\boldsymbol{z})=P(\boldsymbol{x}|\boldsymbol{z}) \text{ whenever } P(\boldsymbol{y},\boldsymbol{z})>0$$

- We write I<**X,Z,Y**>
- Special case: **X** and **Y** are **independent** if

$$P(\boldsymbol{x}|\boldsymbol{y})=P(\boldsymbol{x}) \text{ whenever } P(\boldsymbol{y})>0$$

Chain Rule

- n random variables $X_1,\ldots,X_n$
- Let **U**$=\{X_1,\ldots,X_n\}$
- Joint event **u**$=(x_1,\ldots,x_n)$
- Chain rule:

$$P(\boldsymbol{u}) = P(x_{1,}\ldots,x_n)$$
$$= P(x_n|x_{n-1}\ldots,x_1)\ldots P(x_2|x_1)P(x_1)$$
$$= \prod_{i=1}^{n} P(x_i|x_{i-1}\ldots,x_1)$$

Conditional Independence

- $\boldsymbol{\Pi}_i$ is a subset of $\{X_{i-1},\ldots,X_1\}$ such that
- X_i is conditionally independent of $\{X_{i-1},\ldots,X_1\}\setminus\boldsymbol{\Pi}_i$ given $\boldsymbol{\Pi}_i$

$$P(x_i|x_{i-1}\ldots,x_1) = P(x_i|\boldsymbol{\pi_i})$$

- where $\boldsymbol{\pi}_i$ is a set of values for $\boldsymbol{\Pi}_i$
- $\boldsymbol{\Pi}_i$ **parents** of X_i

Conditional Independence

- Knowing $\boldsymbol{\Pi}_i$ for all i we could write

$$P(\boldsymbol{u}) = P(x_{1,}\ldots,x_n)$$
$$= P(x_n|x_{n-1}\ldots,x_1)\ldots P(x_2|x_1)P(x_1)$$
$$= P(x_n|\boldsymbol{\pi_n})\ldots P(x_2|\boldsymbol{\pi_2})P(x_1|\boldsymbol{\pi_1})$$
$$= \prod_{i=1}^{n} P(x_i|\boldsymbol{\pi_i})$$

Conditional Independence

- In order to compute P(**u**) we have to store

$$P(x_i|\boldsymbol{\pi_i})$$

- for all values x_i and $\boldsymbol{\pi}_i$
- $P(x_i|\boldsymbol{\pi}_i)$: Conditional probability table
- If $\boldsymbol{\Pi}_i$ is much smaller than the set $\{X_{i-1},\ldots,X_1\}$, then we have huge savings
- If k is the maximum number of parents of a variable, then storage is $O(n2^k)$ instead of $O(2^n)$

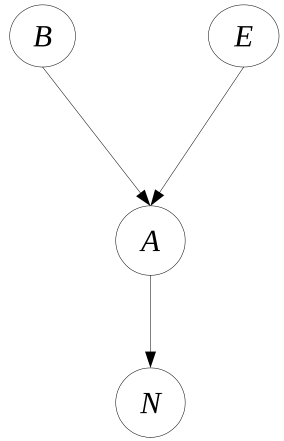
Graphical Representation

- We can represent the conditional independence assertions using a directed graph network with a node per variable
- **Π**$_i$ is the set of parents of X_i
- The graph is acyclic

Example Network

- Variable order: E,B,A,N
- Independences

$$P(e)$$
$$P(b|e)=P(b)$$
$$P(a|b,e)=P(a|b,e)$$
$$P(n|a,b,e)=P(n|a)$$



Bayesian Network

- A Bayesian network [Pearl 85] (BN) B is a couple (G,Θ) where
  - G is a directed acyclic graph (DAG) (V,E) where
    - V is a set of vertices $\{X_1,\dots,X_n\}$
    - E is a set of edges, i.e. A set of couples (X_i,X_j)
    - $<X_1,\dots,X_n>$ is a topological sort of G, i.e. $(X_i,X_j)\in E \Rightarrow i<j$
  - Θ is a set of conditional probability tables (cpts)

$$\{\theta_{x_i|\boldsymbol{\pi}_i} | i=1,\dots,n, x_i \in X_i, \boldsymbol{\pi_i} \in \Pi_i\}$$

  - where **Π**$_i$ is the set of parents of X_i

Bayesian Network

- A BN (G,Θ) **represents** a jpd P iff
  - each variable is independent of its predecessors given its parents in G

$$P(x_i|x_{i-1}\dots,x_1)=P(x_i|\boldsymbol{\pi_i})$$

  - $\theta_{x_i|\boldsymbol{\pi}_i}=P(x_i|\boldsymbol{\pi}_i)$ for all i and $\boldsymbol{\pi}_i$
- In this case

$$P(x_1,\dots,x_n)=\prod_{i=1}^{n} P(x_i|\boldsymbol{\pi_i})$$
$$=\prod_{i=1}^{n} \theta_{x_i|\boldsymbol{\pi}_i}$$

How to Build a Bayesian Network

- Choose an ordering X_1 .. X_n for the variables
- For i = 1 to n:
  - Add X_i node to the network
  - Set $\mathbf{\Pi}_i$ to be a minimal subset of $\{X_1 \ldots X_{i-1}\}$ such that we have conditional independence of X_i and all other members of $\{X_1 \ldots X_{i-1}\}$ given $\mathbf{\Pi}_i$
  - Assign a value to $P(x_i|\boldsymbol{\pi}_i)$ for all the values of x_i and $\boldsymbol{\pi}_i$

Building a Bayesian Network

- Usually the expert consider a variable X as a child of Y if Y is a **direct cause** of X
- Correlation and causality are related but are **not** the same thing
  - See the book [Pearl 00]

Pathfinder system [Suermondt et al. 90]

- Diagnostic system for lymph-node diseases.
- 60 diseases and 100 symptoms and test-results.
- 14,000 probabilities
- Expert consulted to make net.
- 8 hours to determine variables.
- 35 hours for net topology.
- 40 hours for probability table values.

Pathfinder system [Suermondt et al. 90]

- Pathfinder is now outperforming the world experts in diagnosis.
- Being extended to several dozen other medical domains.

Inference with Bayesian Networks

- With a Bayesian Network we save space, do we also save time?
- Do we have to use the formula

$$P(\boldsymbol{q}|\boldsymbol{e})=\frac{\sum_{x,X\in \boldsymbol{V}\setminus \boldsymbol{Q}\setminus \boldsymbol{E}} P(x,\boldsymbol{q},\boldsymbol{e})}{\sum_{x,X\in \boldsymbol{V}\setminus \boldsymbol{E}} P(x,\boldsymbol{e})}$$

- to compute P(**q**|**e**)?

Inference with Bayesian Networks

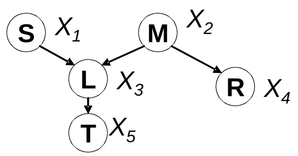
- There are quicker algorithms
  - Exact methods for polytrees
    - Belief propagation
  - Exact methods for general networks
    - Junction tree
    - Variable elimination
  - Approximate methods for general networks:
    - Stochastic sampling
    - Loopy belief propagation
    - Variational methods,

Complexity of Inference

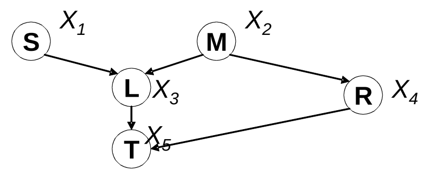
- Exact inference with BN is #P-complete
- #P-complete: a special case of NP-complete problems
  - The answer to a #P-complete problem is the number of solutions to a NP-complete problem

Polytrees

A polytree is a directed acyclic graph in which no two nodes have more than one path between them.



A polytree



Not a polytree

- i.e. There are no cycles in the corresponding undirected graph

Belief Propagation [Pearl 88]

- To compute P(x|**e**) write

$$P(x|\mathbf{e})=\alpha\lambda(x)\pi(x)$$

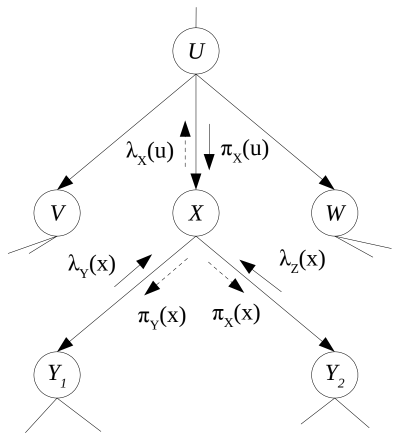
- where α is a normalizing constant and
  - π(x) represents the support to the assertion X=x by the non-descendants of X
  - λ(x) represents the support to the assertion X=x by the descendants of X

Belief Propagation

- Nodes exchange messages with their neighbors
- π(x) and λ(x) are computed from message received respectively from the parents and the children of X
- When a node is activated:
  - It reads the incoming messages
  - It updates π(x) and λ(x)
  - It updates P(x|**e**)
  - It generates the new messages to be sent to their parents and children

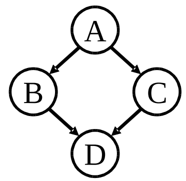
Messages Received

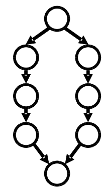
- Node X,
- Parents U_i
- Children Y_j



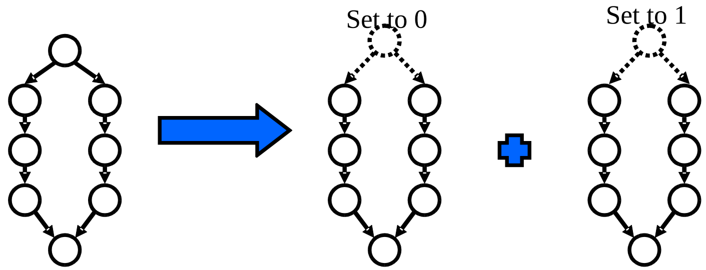
General Networks

- Networks that have a cycle in their undirected version
- Two possibilities
  - Conditioning
  - Clustering



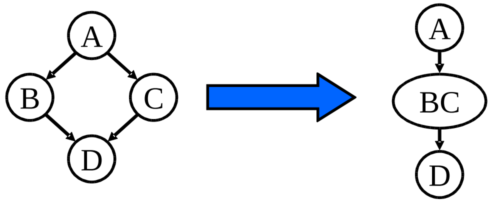


Conditioning



Clustering

- Group together variables so that the resulting network is a polytree and use belief propagation



- Problem: how to find a good clustering?

Join Trees

- Technique for clustering variables
- Steps:
  - Obtain an undirected version of the network
  - Perform a graph operation on it (triangulation)
  - Each clique is a compound variable
  - Add direction to the edges

Junction Tree

- The resulting inference algorithm [Lauritzen, Spiegelhalter 1988] is called
  - Junction tree algorithm (jt), or
  - Clique propagation

Approximate Methods

- Sampling:
  - Generate N samples from BN
  - Count: N_e: samples that satisfy **e**, N_{qe} samples that satisfy **q,e**
  - $P(q|\mathbf{e})=N_{qe}/N_e$
- Loopy belief propagation:
  - bp in networks with cycles
  - Experiments have shown that it converges to good quality solutions

Sampling

- Let $X_1,\ldots,X_n$ be a topological sort of the variables
- For i=1 to n
  - Find parents, if any, of X_i. Call them $X_{p(i,1)}$, $X_{p(i,2)}$, … $X_{p(i,p(i))}$.
  - Recall the values that those parents were randomly given: $x_{p(i,1)}$, $x_{p(i,2)}$, … $x_{p(i,p(i))}$.
  - Look up in the cpt for:

    $P(X_i=x_i \mid X_{p(i,1)}=x_{p(i,1)},X_{p(i,2)}=x_{p(i,2)}\ldots X_{p(i,p(i))}=x_{p(i,p(i))})$
  - Randomly choose x_i according to this probability

Problems in Building BN

- Assessing conditional independence is not always easy for humans
- Usually done on the basis of causal information
- Assigning a number to each cpt entry is also difficult for humans

Problems in Building BN

- Often we do not have an expert but we are given a set of observations $D=\{\mathbf{u}^1,\ldots\mathbf{u}^N\}$
- $\mathbf{u}^j$ is an assignment to all the variables $\mathbf{U}=\{X_1,\ldots,X_n\}$
- How to infer the parameters and/or the structure from D?

Learning

- We want to find a BN over **U** such that the probability of the data P(D) is maximized
- P(D) is also called the **likelihood** of the data
- We assume that all the samples are **independent and identically distributed** (iid) so

$$P(D)=\prod_{i}^{N} P(\boldsymbol{u}^{i})$$

- Often the natural log of P(D) (**log likelihood**) is considered

$$\log P(D)=\sum_{i}^{N} \log P(\boldsymbol{u}^{i})$$

Learning BN

- Tasks
  - Computing the parameters given a fixed structure or
  - finding the structure and the parameters
- Properties of data:
  - complete data: in each data vectors $\mathbf{u}^j$, the values of all the variables are observed
  - incomplete data

Parameter Learning from Complete Data

- Parameters to be learned

$$\theta_{x_i|\boldsymbol{\pi}_i}=P(x_i|\boldsymbol{\pi}_i)$$

- for all x_i, $\boldsymbol{\pi}_i$, i=1,...,n
- The values of the parameters that maximize the likelihood can be computed in closed form

Maximum Likelihood Parameters

- Given by relative frequency
- If $N_{\mathbf{y}}$ be the number of vectors of D where **Y**=**y**.

$$\theta_{x_i|\boldsymbol{\pi}_i}=\frac{N_{x_i,\boldsymbol{\pi}_i}}{N_{\boldsymbol{\pi}_i}}$$

- Counting: for each i, for each value $\boldsymbol{\pi}_i$ we must collect

$$C_{\boldsymbol{\pi}_i}=\langle N_{x_i^1,\boldsymbol{\pi}_i},\ldots,N_{x_i^{v(i)},\boldsymbol{\pi}_i}\rangle$$

- where v(i) is the number of values of X_i

Structure Learning from Complete Data

- Perform a local search in the space of possible structures
- HGC algorithm [Heckerman, Geiger, Chickering 95]:
  - Start with a structure BestG' (possibly empty)
  - Repeat
    - BestG=BestG'
    - Let Ref={G'|G' is obtained from BestG' by adding, deleting or reversing an arc}
    - Let BestG'=argmax$_{G'}$ {score(G')|G' $\in$ Ref}
  - while score(BestG')-score(BestG)>0

Structure Score

$$score(G)=P(D|G)$$

$$P(D|G)=\int \rho(D,\Theta|G)\,d\Theta$$
$$=\int P(D|\Theta,G)\rho(\Theta)\,d\Theta$$

- where

$$\rho(\Theta)=\prod_{i,\pi_i}\rho(\theta_{\pi_i})$$
$$\theta_{\pi_i}=\langle\theta_{x_i^1|\pi_i},\ldots,\theta_{x_i^{v(i)}|\pi_i}\rangle$$

- and $\rho(\theta_{\pi i})$ is the prior density of the vector $\theta_{\pi i}$

Prior Density of the Parameters

- A common choice for the form of the prior density is the **Dirichlet probability density**
- In this case $\rho(\theta_{\pi i})$ is described by v(i) parameters

$$C'_{\pi_i}=\langle N'_{x_i^1,\pi_i},\ldots,N'_{x_i^{v(i)},\pi_i}\rangle$$

- Prior counts: it is as if we had previously observed some data on which the counts are $N'_{xi,\pi i}$

Structure Score

- If the priors for the parameters are Dirichlet, then the score is called BD (for Bayesian Dirichlet) and

$$BD(G)=\sum_i BD_i(G)$$

- where $BD_i(G)$ depends only on C_i and C'_i, the counts for the family of X_i

$$C_i\ =\langle C_{\pi_i^1},\ldots,C_{\pi_i^{p(i)}}\rangle$$
$$C'_i=\langle C'_{\pi_i^1},\ldots,C'_{\pi_i^{p(i)}}\rangle$$

Structure Score

- BD(G) is **decomposable**:
  - It can be computed independently for each family
- Each edge operation involves
  - 1 family (addition, deletion) or
  - 2 families (reversal)
- BD(G') can be quickly computed from BD(BestG) by changing only the score of the affected families

Parameter Learning from Incomplete Data

- The maximum likelihood parameters cannot be computed in closed form
- An iterative algorithm is necessary: the EM algorithm
- Finds a (possibly) local maximum of the likelihood

EM Algorithm

- Initialize the parameters at random Θ
- Repeat
  - Expectation step:
    - compute the probability of each value of the missing attributes using (G,Θ) and inference
    - Obtain a new dataset D' by completing D according to the probabilities computed above
  - Compute Θ by maximum likelihood on D'
    - Relative frequency

Structure Learning from Incomplete Data

- There is no decomposable score
- HGC would not be efficient
- Structural EM:
  - Start with a structure BestG' (possibly empty)
  - Repeat
    - BestG=BestG'
    - Compute the parameters of BestG with EM
    - Optimize a lower bound of the likelihood of the observed data
    - Let BestG' the optimum
  - Until no improvement

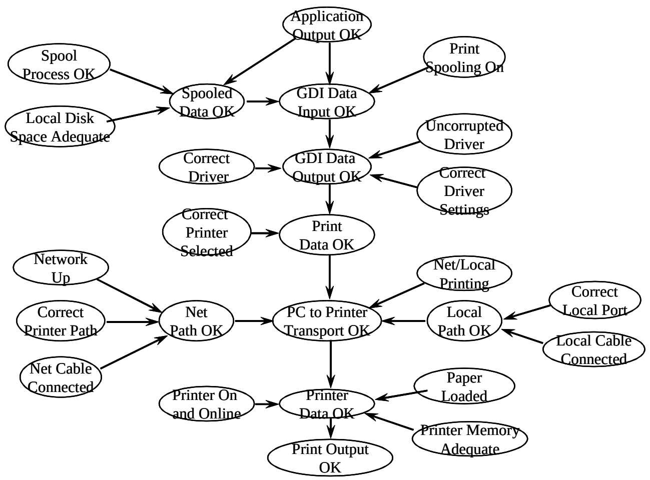
Applications of BN

- Monitoring of emergency care patients
- Model of barley crops yield.
- Diagnosis of carpal tunnel syndrome
- Insulin dose adjustment (DBN) in diabetes .
- Predicting hails in northern Colorado.
- Evaluating insurance applications

Applications of BN

- Deciding on the amount of fungicides to be used against attack of mildew in wheat.
- Assisting experts of electromyography.
- Pedigree of breeding pigs.
- Modeling the biological processes of a water purification plant.
- Printer troubleshooting (Microsoft Windows)

Printer Troubleshooting (Windows 95)



Applications

- Office Assistant in MS Office ("smiley face")
  - Bayesian network based free-text help facility
  - help based on past experience (keyboard/mouse use) and task user is doing currently

Markov Networks (MN)

- Approach alternative to BN
- Undirected graph
- Conditional independence represented by graph separation
- Probability distribution as the product of a set of **potentials** (functions of a subset of variables) divided by a normalization constant
- One potential per clique

Markov Network

- Inference:
  - Algorithms similar to those for BN (bp, ct, ve, ss..)
  - Same complexity
- MN can represent some independences that BN can not represent and vice versa
- Advantage: we do not have to avoid cycles
- Disadvantage: MN parameters are more difficult to interpret

Combination of Logic and Probability

- BN are not able to deal with domains containing multiple entities connected by complex relationships
- Logic is not able to represent uncertainty efficiently
- Combination: active research area with many different proposals
- The most common approach is to design a new language and then provide a translation into BN or MN for defining the semantics, performing inference and learning

Some Logical-Probabilitstic Languages

- Probabilistic Relational Models →BN
- Markov Logic Network →MN
- Bayesian Logic Programs →BN
- Logic Programs with Annotated Disjunctions→BN
- Relational Markov Networks →MN
- CLP(BN) →BN

CLP(BN) [Costa et al 03]

- Based on Prolog
- Variables in a CLP(BN) program can be random
- Their values, parents and CPTs are defined with the program
- To answer a query with uninstantiated random variables, CLP(BN) builds a BN and performs inference
- The answer will be a probability distribution for the variables

Example

```
.....
registration_grade(Key, Grade):-
registration(Key, CKey, SKey),
course_difficulty(CKey, Dif),
student_intelligence(SKey, Int),
{ Grade = grade(Key) with
 p([a,b,c,d],
%h h  h m  h l  m h  m m  m l  l h  l m  l l
[0.20,0.70,0.85,0.10,0.20,0.50,0.01,0.05,0.10,
 0.60,0.25,0.12,0.30,0.60,0.35,0.04,0.15,0.40,
 0.15,0.04,0.02,0.40,0.15,0.12,0.50,0.60,0.40,
 0.05,0.01,0.01,0.20,0.05,0.03,0.45,0.20,0.10 ],
[Int,Dif]))
}.
.....
```


Inference

```
   ?- [school_32].
   ?- registration_grade(r0,G).
p(G=a)=0.4115,
p(G=b)=0.356,
p(G=c)=0.16575,
p(G=d)=0.06675 ?
?- registration_grade(r0,G),
   student_intelligence(s0,h).
p(G=a)=0.6125,
p(G=b)=0.305,
p(G=c)=0.0625,
p(G=d)=0.02 ?
```


Availability

- CLP(BN) is included in Yap prolog
- http://www.dcc.fc.up.pt/~vsc/Yap/
- It can use either junction tree or variable elimination for inference

Logic Programs with Annotated Disjunction

- [Vennekens et al. 04]
- Minimal extension of logic programming to allow the representation of uncertainy
- Clauses of the form

$$h_1:\alpha_1 \;;\; \ldots \;;\; h_n:\alpha_n \;\text{:-}\; b_1,\ldots b_m$$

- where h_i are atoms, b_i are literals and α_1 are probabilities such that

$$\sum_{i=1}^{n} \alpha_i \leq 1$$

Semantics

- Each clause can be seen as an experiment: if $b_1,\ldots b_m$ is true then h_i is true with probability α_i or no h_i is true with probability $1\text{-}\Sigma_i\alpha_i$
- Each ground atom is seen as random variable with values true and false
- We want to assign probabilities to queries (conjunctions of ground atoms), possibly conditioned on some evidence

Semantics

- Given an LPAD T, generate its grounding T'
- An instance of T is a normal logic program obtained by selecting one head from each clause of T'
- The probability of an instance is obtained by multiplying the probability of each head selected
- The probability of a query Q is given by the sum of the probabilities of the instances that have Q as consequence

Example

```
heads(Coin):0.5 ; tails(Coin):0.5 :-
  toss(Coin), \+ biased(Coin).
heads(Coin):0.6 ; tails(Coin):0.4 :-
  toss(Coin),  biased(Coin).
biased(Coin):0.1 ; fair(Coin):0.9.
toss(coin).
```

P(heads(coin))=0.51

P(heads(coin)|biased(coin))=0.6

Conversion to Bayesian Networks

- An LPAD can be converted to a BN that has
  - One boolean variable per ground atom
  - One variable ch_r per ground clause r, with the ground atoms in the head plus null as values
- The dependencies are defined as follows:
  - Ground atom a depends on all the clause variables that have a in the head
  - The CPT assign probability 1 to a if at least one parent is equal to a and 0 otherwise

Conversion to Bayesian Networks

- ch_r depends on the variables that appear in the body of r
- CPT:
  - $P(ch_r=h_i)=\alpha_i$, $P(ch_r=null)=1-\Sigma_i\alpha_i$ if the body is true
  - $P(ch_r=null)=1$ if the body is false

Inference with LPADs

- Convert to BN and use BN inference
  - Problem: the grounding may be very large
- Compute all the possible derivations and compute the probability that one of these derivations is possible [Riguzzi 07]
- Suite of reasoning tools for LPADs: cplint

  http://www.ing.unife.it/software/cplint/
- It is included in the CVS version of Yap

Learning LPADs

- Data D: set of interpretations (i.e. sets of ground atoms),
- Task: find the parameters of an LPAD that maximize the likelihood of D:
- Task: find the parameters and the structure of an LPAD that maximize the likelihood of D

Learning Parameters

- ME-compliant LPAD: every couple of ground clauses that share an atom in the head have mutually exclusive bodies
- If an LPAD is ME-compliant then the parameters can be computed in closed form as a ratio of counts [Riguzzi 04]

$$\alpha_i = P(h_i | body)$$

- Otherwise [Blookeel, Meert 06]
  - Convert the LPAD to a BN
  - Use EM since the ch_r variables are unobserved in D

Learning the Structure

- If the LPAD is ME-compliant then the structure can be learned by solving a mixed integer programming problem
  - ALLPAD system [Riguzzi 08]
- Otherwise [Blockeel, Meert, 07]
  - Use Structural EM to learn a BN
  - Convert to LPAD

BN Software

- List of BN software http://www.cs.ubc.ca/~murphyk/Software/bnsoft.html\
- BNT: inference and learning, Matlab, open source
- MSBNx: inference, by Microsoft, free closed source
- OpenBayes: inference and learning, Python, open source
- BNJ: inference and learning, Java, open source
- Weka: learning, Java, open source

Resources

- Probabilistic Reasoning in Intelligent Systems by Judea Pearl. Morgan Kaufmann: 1998.
- Probabilistic Reasoning in Expert Systems by Richard Neapolitan. Wiley: 1990.
- List of BN Models and Datasets http://www.cs.huji.ac.il/labs/compbio/Repository/

Acknowledgments

- Some slides from
  - Andrew Moore's tutorials http://www.autonlab.org/tutorials/
  - Irina Rish and Moninder Singh's tutorial http://www.research.ibm.com/people/r/rish/

References

- [Pearl 85] Pearl, J., "Bayesian Networks: a Model of Self-Activated Memory for Evidential Reasoning," UCLA CS Technical Report 850021, Proceedings, Cognitive Science Society, UC Irvine, 329-334, August 15-17, 1985.
- [Pearl 00] Pearl, J., Causality: Models, Reasoning, and Inference, Cambridge University Press, 2000
- [Suermondt et al. 90] Henri Jacques Suermondt, Gregory F. Cooper, David Heckerman, “A combination of cutset conditioning with clique-tree propagation in the Pathfinder system”, UAI '90.

References

- [Pearl 88] Judea Pearl, Probabilistic Reasoning in Intelligent Systems, Morgan Kaufmann: 1998.
- [Lauritzen, Spiegelhalter 1988]
- [Heckerman, Geiger, Chickering 95] D. Heckerman, D. Geiger, D. M. Chickering: “Learning Bayesian Networks: The Combination of Knowledge and Statistical Data”, Machine Learning, 20(3), 1995
- [Costa et al 03] V/ Santos Costa, D. Page, M. Qazi, J. Cussens: “CLP(BN): Constraint Logic Programming for Probabilistic Knowledge”, UAI03.

References

- [Vennekens et al. 04] J.Vennekens, S. Verbaeten and M. Bruynooghe, “Logic programs with annotated disjunctions”, ICLP04
- [Riguzzi 07] F. Riguzzi. “A top down interpreter for lpad and cp-logic”. In Proceedings of the 10th Congress of the Italian Association for Artificial Intelligence, number 4733 in Lecture Notes in Artificial Intelligence, Springer, 2007.
- [Riguzzi 04] F. Riguzzi. “Learning logic programs with annotated disjunctions”, ILP04.

References

- [Blookeel, Meert 06] H. Blockeel, W. Meert: "Towards Learning Non-recursive LPADs by Transforming Them into Bayesian Networks", ILP 2006:
- [Riguzzi 08] F. Riguzzi, "ALLPAD: Approximate learning of logic programs with annotated disjunctions". Machine Learning, 70(2-3), 2008.
- [Blookeel, Meert 07] W. Meert, J. Struyf, H. Blockeel, "Learning Ground CP-logic Theories by means of Bayesian Network Techniques", MRDM07