# Graphical probabilistic models – learning from data

# Jiří Kléma

Department of Cybernetics, FEE, CTU at Prague



http://ida.felk.cvut.cz

# Agenda

- Motivation for learning from data
  - knowledge is hard to obtain  $\times$  data of sufficient size often at hand,
- structure of training data
  - frequency table is commonly sufficient,
  - incomplete data make learning harder,

#### parameter learning

- easier (sub)task,
- MLE algorithm (+ EM for incomplete data),
- data quantity demonstration of requirements
  - \* original (known) network  $\rightarrow$  training samples  $\rightarrow$  learned network  $\rightarrow$  comparison with the original one,

#### structure learning

- more difficult task,
- structure selection criteria? likelihood, MAP score, BIC,
- naïve approach, K2 and MCMC algorithms,
- illustrative examples.

#### Learning Bayesian networks from data

format of training data?

 $d_1 =$ 

 $d_2 =$ 

 $d_M =$ 

- sample set D contains M samples = concurrent observations of all the variables,
- FAMILY example:  $d_m = \{FO_m, BP_m, LO_m, DO_m, HB_m\}$ ,  $m = 1 \dots M$ ,
- no missing values concerned yet for simplicity,
- frequency table (hypercube) provides sufficient statistics (representation)
  - gives the number of samples with particular configuration (frequency over sample space),
  - $-2^5$  entries  $N(\{fo, bp, do, lo, hb\})$ , ...,  $N(\{\neg fo, \neg bp, \neg do, \neg lo, \neg hb\})$ ,
  - representation close to the joint probability distribution.

$$\{fo_1, \neg bp_1, \neg lo_1, do_1, \neg hb_1\} \\ \{\neg fo_2, \neg bp_2, \neg lo_2, do_2, hb_2\} \\ \cdot \\ \{\neg fo_M, bp_M, \neg lo_M, do_M, hb_M\}$$

-bp

106

45

4

349 71

> 30 2

233

### Learning Bayesian network parameters from data

- network structure is known, we search for CPTs in the individual nodes,
- maximum likelihood estimate (MLE) of unknown parameters  $\Theta$ 
  - FAMILY example

$$L(\Theta:D) = \prod_{m=1}^{M} Pr(d_m:\Theta) = \prod_{m=1}^{M} Pr(FO_m, BP_m, LO_m, DO_m, HB_m:\Theta) = \prod_{m=1}^{M} Pr(FO_m:\Theta) Pr(BP_m:\Theta) Pr(LO_m | FO_m:\Theta) \dots Pr(HB_m | DO_m:\Theta)$$

- for general Bayesian network

$$L(\Theta:D) = \prod_{m=1}^{M} Pr(d_m:\Theta) = \prod_{m=1}^{M} Pr(P_{1m}, P_{2m}, \dots, P_{nm}:\Theta) =$$
$$= \prod_{j=1}^{n} \prod_{m=1}^{M} Pr(P_j | parents(P_j):\Theta_j) = \prod_{j=1}^{n} L_j(\Theta_j:D)$$

• under the assumption of independence of parameters, likelihood can be decomposed

- contribution of each network node  $L_j(\Theta_j : D)$  is determined (maximized) independently.

#### Learning Bayesian network parameters from data

- the optimization task:  $\widehat{\Theta_j} = \underset{\Theta}{\operatorname{arg\,max}} L_j(\Theta_j : D)$  is solved for each node,
- let us demonstrate for FO node, where  $\Theta_{FO} = \{Pr(fo)\}$ 
  - let N(fo) be the number of samples, where  $FO_j = TRUE$
  - $L_{FO}$  is maximized by putting its first derivative equal to 0

$$L_{FO}(\Theta_{FO}:D) = \prod_{m=1}^{M} Pr(FO:\Theta_{FO}) = Pr(fo)^{N(fo)}(1 - Pr(fo))^{M-N(fo)}$$
$$\frac{\partial L_{FO}(Pr(fo):D)}{\partial \Pr(fo)} = 0 \to Pr(fo) = \frac{N(fo)}{M}$$

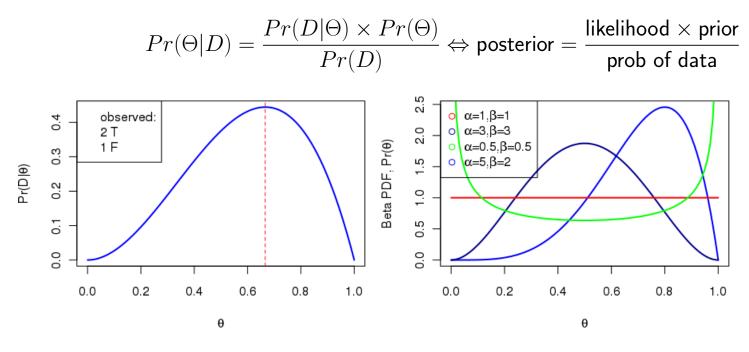
• the generalized formula for ML parameter estimation is intuitively obvious

$$\widehat{\theta}_{P_j | parents(P_j)} = \frac{N(P_j, parents(P_j))}{N(parents(P_j))} \approx Pr(P_j | parents(Pj))$$

- however, this estimate is imprecise/impossible for sparse/incomplete data
  - sparse data  $\rightarrow$  Dirichlet priors and maximum a posteriori (MAP) probability method,
  - missing data  $\rightarrow$  Monte-Carlo sampling, or
    - $\rightarrow$  EM optimization of multimodal likelihood function.

# Parameter learning from a small number of observations

- ill-posed problem
  - overfitting, division by zero, zero probabilities learned,
- regularization
  - introducing additional information in order to resolve an ill-posed problem,
  - Bayesian learning makes use of prior probability



• MAP estimate of parameters:  $\hat{\theta}_{P_j|parents(P_j)} = \frac{N(P_j, parents(P_j)) + \alpha - 1}{N(parents(P_j)) + \alpha + \beta - 2}$ 

# Parameter learning from incomplete data

- missing values completely at random
  - the simplest option independent of variable states, no hidden parameters used,
- it is not advisable to ignore the missing values
  - loses existing observations as well,
- MLE combined with EM algorithm:
  - 1. initialize network parameters (typically using available training data or randomly),
  - 2. E step: take the existing network and compute the missing values (inference),
  - 3. M step: modify the network parameters according to the current complete observations, use MLE,
  - 4. repeat steps 2 and 3
    - (a) for the given prior number of iterations (in this experiment 10),
    - (b) until convergence of MLE criterion (log L change between consecutive steps < 0.001).

# **Parameter learning from data – illustration of convergence**

- 1. take existing (original) network and generate training data (a sample set)
  - FAMILY network, consider different *M* values (sample set sizes),
  - in which way to generate the data?
    - no evidence, thus forward sampling, see inference
    - Gibbs sampling is also possible,
- 2. randomize quantitative network parameters
  - the network structure is preserved,
  - the original CPTs lost,
- 3. parameter values are learned from training data
  - complete observations maximum likelihood estimate (MLE),
  - incomplete observations combination of MLE and EM algorithm,
- 4. compare the original and learned CPTs for different sample set sizes M
  - why is it easier to estimate Pr(fo) then Pr(do|fo, bp)? see graphs ...

#### **Parameter learning from data – complete observations**

• What is the probability that family is out?

$$-Pr(fo) = ?$$

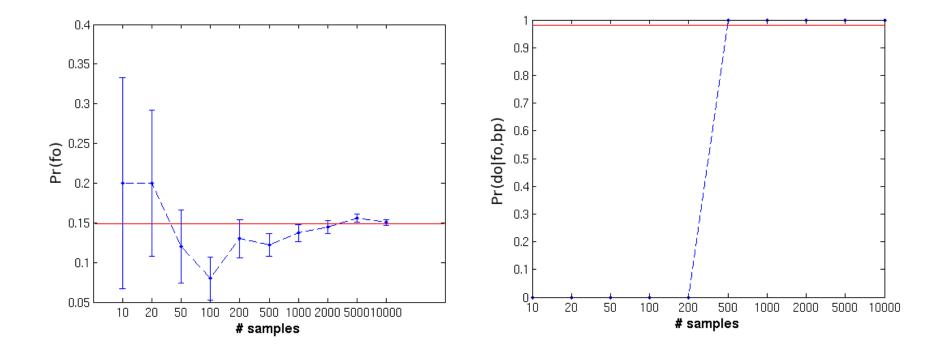
• all samples can be used . . .

$$- Pr(fo) = \frac{\sum_{m=1}^{M} \delta(FO^m, fo)}{M}$$

- What is the dog out prob given fo and bp? - Pr(do|fo, bp) = ?
- Condition is met only in 1.5  $^{0}\!/_{00}$  of samples.

$$- Pr(fo) = 0.15$$
,  $Pr(bp) = 0.01$ ,

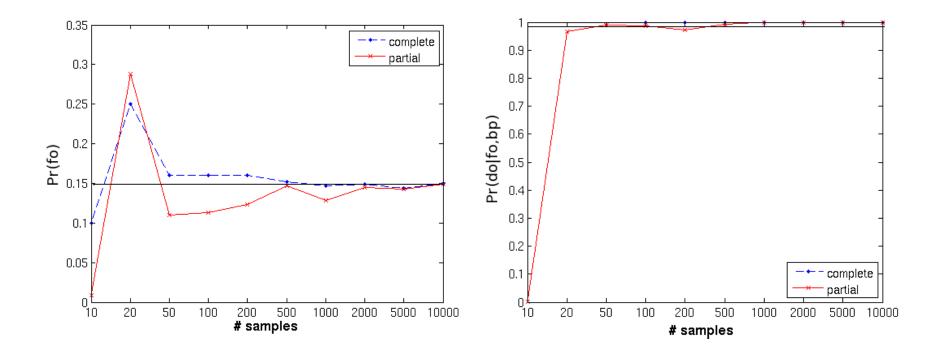
-FO and BP independent variables.



# Parameter learning from data – incomplete observations (50% loss)

- What is the probability that family is out?
  - -Pr(fo) = ?
- Incomplete data = less information
  - considerably longer computational time, comparison is inconclusive.
  - the final estimate "a bit less exact only".

- What is the dog out prob given *fo* and *bp*? -Pr(do|fo, bp) = ?
- Incomplete data = less information



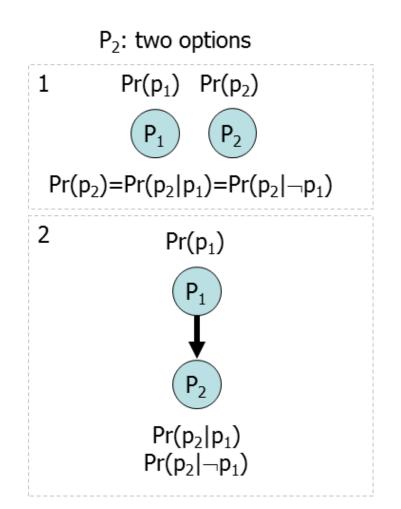
### **Structure learning – naïve approach**

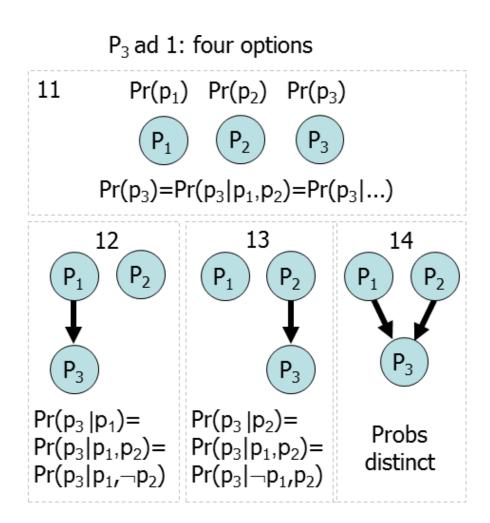
- two steps are sufficient to construct the network:
  - 1. define a sort of n variables,
  - 2. gradually find subsets of variables that satisfy conditional independence relationship  $Pr(P_{j+1}|P_1,\ldots,P_j) = Pr(P_{j+1}|parents(P_{j+1}))$ ,  $parents(P_{j+1}) \subseteq \{P_1,\ldots,P_j\}$ ,
- the algorithm can be illustrated on a simple three variable example:
  - 1. select a permutation  $\pi$ :  $\pi(P_1) = 1$ ,  $\pi(P_2) = 2$  a  $\pi(P_3) = 3$ ,
  - 2. gradually build a network, add nodes one by one, conditional independence test underlies the local decision.

### **Structure learning – naïve approach**

- two steps are sufficient to construct the network:
  - 1. define a sort of n variables,
  - 2. gradually find subsets of variables that satisfy conditional independence relationship  $Pr(P_{j+1}|P_1,\ldots,P_j) = Pr(P_{j+1}|parents(P_{j+1}))$ ,  $parents(P_{j+1}) \subseteq \{P_1,\ldots,P_j\}$ ,
- the algorithm can be illustrated on a simple three variable example:
  - 1. select a permutation  $\pi$ :  $\pi(P_1) = 1$ ,  $\pi(P_2) = 2$  a  $\pi(P_3) = 3$ ,
  - 2. gradually build a network, add nodes one by one, conditional independence test underlies the local decision.
- cannot be implemented in this easy form:
  - variable ordering influences the resulting network there is **n!** distinct permutations
    - \* the complete graph can originate from an improper ordering, however, all permutations cannot be checked,
  - independence tests also non-trivial
    - \* for binary variables definitely  $\mathcal{O}(2^n)$  operations per single permutation,
    - \* among others, the conditional probability  $Pr(P_n|P_1, \ldots, P_{n-1})$  needs to be enumerated.

# Structure learning – naïve approach





#### **Structure learning – more feasible implementations**

- score-based learning, maximizes an evaluation function
  - the function quantifies how well a structure matches the data,
- straightforward likelihood function selects the fully connected network (complete graph)
  - the more parameters, the better match with data,
  - results in overfitting improper when comparing structures of different size,

$$\log L(G:D) = \log \prod_{m=1}^{M} Pr(d_m:G) = M \sum_{j=1}^{n} (I(P_j:parents(P_j)^G) - H(P_j))$$

evaluation function often based on Bayesian score that stems from posterior probability

$$Pr(G|D) = \frac{Pr(D|G)Pr(G)}{Pr(D)} \rightarrow \log Pr(G|D) = \log Pr(D|G) + \log Pr(G) + c$$

- unlike MLE, it integrates over all parametrizations of given structure

$$Pr(D|G) = \int Pr(D|G, \Theta_G) \times Pr(\Theta_G|G)d\Theta$$

- MLE concerns solely the best parametrization

$$L(G:D) = Pr(D|G,\widehat{\Theta_G})$$

#### **Structure learning – more feasible implementations**

- Bayesian Information Criterion (BIC) represents another frequent evaluation function
  - a heuristic criterion, easier to compute than the Bayesian one,
  - a MDL principle analogy the best model is both compact and accurate,
  - let us have:  $q_i$  ... the number of unique instantiations of  $P_i$  parents,

 $r_i$  ... the number of distinct  $P_i$  values,

- then, a network has:  $K = \sum_{i=1}^n q_i(r_i - 1)$  independent parameters,

$$BIC = -\frac{K}{2}\log_2 M + \log_2 L(G:D) = -\frac{K}{2}\log_2 M - M\sum_{i=1}^n H(P_i|parents(P_i)^G)$$

- first addend: network complexity penalty (K  $\uparrow$  BIC  $\downarrow$ ),
- second addend: network likelihood

(mutual information between nodes and their parents  $\uparrow H(|) \downarrow BIC \uparrow$ ),

— how to enumerate conditional entropy?

\*  $N_{ij}$  ... the number of samples, where  $parents(P_i)$  take the j-th instantiation of values, \*  $N_{ijk}$  ... the number of samples, where  $P_i$  takes the k-th value and  $parents(P_i)$  the j-th instantiation of values,

$$H(P_i | parents(P_i)^G) = -\sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \frac{N_{ij}}{M} \frac{N_{ijk}}{N_{ij}} \log_2 \frac{N_{ijk}}{N_{ij}} = -\sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \frac{N_{ijk}}{M} \log_2 \frac{N_{ijk}}{N_{ij}}$$

# **Structure learning – more feasible implementations**

- however, no evaluation function can be applied to all  $2^{n^2}$  candidate graphs,
- heuristics and metaheuristics known for difficult tasks need to be employed
  - metaheuristic example local search
    - \* it starts with a given network (empty, expert's, random),
    - \* it construct all the "near" networks, evaluates them and goes to the best of them,
    - \* it repeats the previous step if the local change increases score, otherwise it stops,
  - auxiliary heuristics examples
    - \* definition of "near" network,
    - \* how to avoid getting stuck in local minima or on plateaux
      - $\cdot$  random restarts, simulated annealing, TABU search.

# **Structure learning – K2 algorithm**

• Cooper and Herskovitz (1992), it approaches the naïve approach mentioned above,

advantage

- $\text{ complexity is } \mathcal{O}(m, u^2, n^2, r), \ u \leq n \rightarrow \mathcal{O}(m, n^4, r)$ 
  - $* m \ldots$  the number of samples,  $n \ldots$  the number of variables,
  - $* r \dots$  max number of distinct variable values,  $u \dots$  max number of parents,

disadvantages

- topological sort of network variables  $\pi$  must be given/known,
- greedy search results in locally optimal solution.
- $\hfill \ensuremath{\,\bullet\)}$  it expresses the prob Pr(G,D) as the following function

$$g(P_i, parents(P_i)) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!$$

- $-q_i \dots$  number of unique instantiations of  $parents(P_i)$ ,  $r_i \dots$  number of distinct  $P_i$  values,
- $N_{ij}$  ... number of samples, where  $parents(P_i)$  take j-th instantiation of values,
- $N_{ijk}$  ... number of samples, where  $P_i$  takes k-th value and  $parents(P_i)$  j-th instantiation of values,
- **separable** criterion it can be computed node by node.

#### **Structure learning – K2 algorithm**

- algorithm K2  $(\pi, u, D)$ :
  - for i=1:n % follow the topological sort of variables  $\pi$

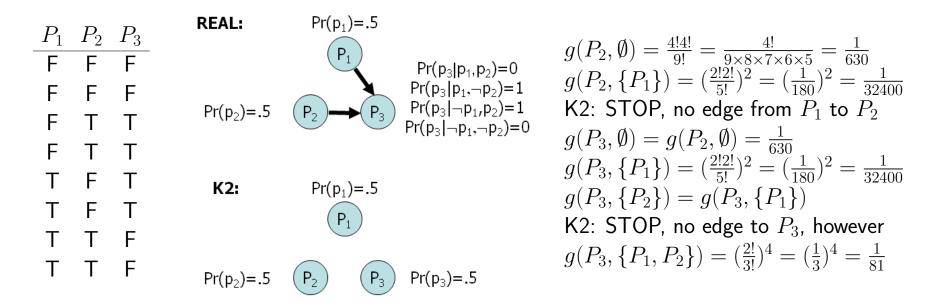
parents  $(P_{\pi_i}) = \emptyset$  % in the beginning, the set of parents is always empty  $G_{old} = g(P_{\pi_i}, \text{parents}(P_{\pi_i}))$  % initialize the node value while  $|\text{parents}(P_{\pi_i})| \le u$  % the number of parents must not exceed u

$$j^* = \underset{j=1\dots i-1, P_{\pi_j} \notin parents(P_{\pi_i})}{\arg\max} g(P_{\pi_i}, parents(P_{\pi_i}) \cup P_{\pi_j})$$

% 
$$P_{\pi_j^*}$$
 is the parent maximizing the value of g  
% the parent must have a lower topological index -- by definition  
% omit the candidates already belonging to the set of parents  
 $G_{new}=g(P_{\pi_i}, \text{parents}(P_{\pi_i}) \cup P_{\pi_j^*})$   
if  $G_{new} > G_{old}$  then  
 $G_{old} = G_{new}$   
parents $(P_{\pi_i})=\text{parents}(P_{\pi_i} \cup P_{\pi_j^*})$   
else

STOP % the node value cannot be further improved, stop its processing

• let us have binary variables  $P_1$ ,  $P_2$ ,  $P_3$ , let  $\pi = \{1,2,3\}$  and D is given in the table

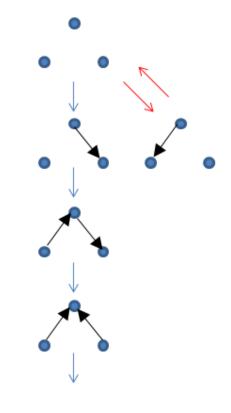


minor improvements

- apply K2 and K2Reverse and take the better solution
  - \* K2Reverse starts with the complete graph and greedily deletes edges,
  - \* solves the particular problem shown above, but not a general solution,
- randomly restart the algorithm (various node orderings and initial graphs).

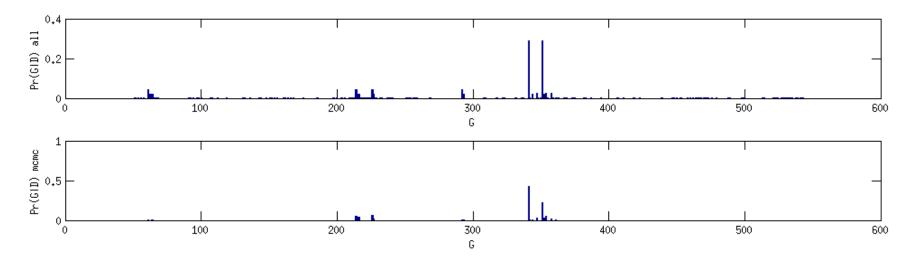
# **Structure learning – MCMC approach**

- MCMC = Markov chain Monte-Carlo (for meaning see Gibbs sampling),
- applies Metropolis-Hastings (MH) algorithm to search the candidate graph/network space
  - 1. take an initial graph G
    - user-defined/informed, random, empty with no edges,
  - 2. evaluate the graph  ${\cal P}({\cal G})$ 
    - use samples, apply criteria such as BIC or Bayesian,
  - 3. generate a "neighbor"  ${\cal S}$  of the given graph  ${\cal G}$ 
    - insert/remove an edge, change edge direction,
    - check the graph acyclicity constraint,
    - prob of transition from G to S is function of  $Q(G,S)\mbox{,}$
  - 4. evaluate the neighbor graph P(S),
  - 5. accept or reject the transition to  $\boldsymbol{S}$ 
    - generate  $\alpha$  from U(0,1) (uniform distribution),
    - if  $\alpha < \frac{P(S)Q(G,S)}{P(G)Q(S,G)}$  then accept the transition  $G \to S$ ,
  - 6. repeat steps 3–5 until convergence or the given number of iterations.



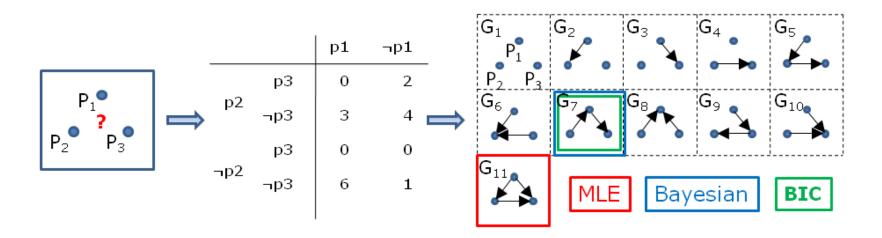
# **Structure learning – MCMC approach**

- graph frequency helps to assume on their posterior probability
  - a sequence beginning is ignored for random inits,
- the sequence of graphs can be used both for point and **Bayesian** estimation
  - point estimation e.g., only the network with the highest score is concerned (MAP),
  - Bayesian estimation all the networks concerned and weighted by their score,
- convergence (frequency proportional to the real score)
  - theoretically converges in polynomial time wrt size of graph space,
  - practically difficult for domains with more than 10 variables.



# **Structure learning – 3DAG example**

- let us concern a 3 node trial network and generate 16 samples of it,
- score a member of each Markov equivalence class (complete search, 11 graphs),
- apply 3 distinct criteria (max likelihood, Bayesian MAP and BIC) to identify the best model.



• G<sub>1</sub> gradually evaluated by three criteria:

- likelihood: ML parameters first  $Pr(p_1) = Pr(p_2) = \frac{9}{16}$ ,  $Pr(p_3) = \frac{1}{8}$ 

$$\ln L(G_1:D) = \sum_{m=1}^{16} Pr(d_m:G_1) =$$
$$= 2\ln\left(\frac{7}{16}\frac{9}{168}\right) + 3\ln\left(\frac{9}{168}\frac{9}{168}\right) + 10\ln\left(\frac{9}{168}\frac{7}{168}\right) + \ln\left(\frac{7}{168}\frac{7}{168}\right) = -27.96$$

# **Structure learning – 3DAG example**

- the identical likelihood value can also be reached through conditional entropy

$$\ln L(G_1:D) = -M \sum_{i=1}^{3} H(P_i | parents(P_i)^{G_1}) =$$
$$= -16 \left[ -2 \left( \frac{9}{16} \ln \frac{9}{16} + \frac{7}{16} \ln \frac{7}{16} \right) - \left( \frac{1}{8} \ln \frac{1}{8} + \frac{7}{8} \ln \frac{7}{8} \right) \right] = -27.96$$

 BIC determined by subtracting the network complexity penalty from the value of network likelihood

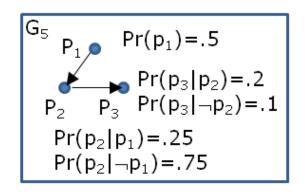
$$BIC(G_1:D) = -\frac{K}{2}\ln M + \ln L(G_1:D) = -\frac{3}{2}\ln 16 - 27.96 = -32.12$$

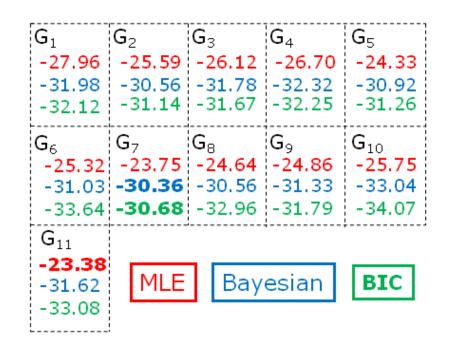
- Bayesian score

$$\ln \Pr(D|G_1) = \ln \prod_{i=1}^{3} g(P_i, parents(P_i)^{G_1}) = \sum_{i=1}^{3} \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} \ln \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} N_{ijk}! = 2(-\ln 17! + \ln 9! + \ln 7!) - \ln 17! + \ln 2! + \ln 14! = -31.98$$

Natural logarithm is applied to match Matlab BN Toolbox.

Logarithm base change does not change ordering of model evaluations.





- none of three criteria identified the correct graph class
  - MLE overfits the sample set as expected,
  - BIC and MAP suffer from insufficient data (a too small sample set).

# **Summary**

• Estimation of (quantitative) Bayesian network parameters

- relatively easy ML or MAP estimate,
  - \* they agree for large samples, differ for small ones, MAP preferable when a prior distribution exists,
- gets more difficult with small or incomplete sample sets
  - \* prior knowledge resp. iterative EM refinement (parameters  $\leftrightarrow$  observations),
- BN structure discovery as score-based learning
  - several scores to evaluate how well a structure matches the data
    - \* likelihood, resp. log likelihood (two ways to compute available)  $\rightarrow$  bad idea, overfits,
    - \* Bayesian score, BIC based on likelihood,
    - \* other options among others local conditional independence tests,
  - the space of candidate structures is huge
    - \* the space cannot be exhaustively searched, i.e., the scores computed for all candidates,
    - \* consequently, even the naïve approach cannot be considered,
    - \* K2 a greedy, locally optimal search,
    - \* MCMC a stochastic search similar to simulated annealing.

- Murphy: A Brief Introduction to Graphical Models and Bayesian Networks.
  - a practical overview from the author of BN toolbox,
  - http://www.cs.ubc.ca/~murphyk/Bayes/bayes.html#learn,
- Friedman, Koller: Learning Bayesian Networks from Data.
  - Neural Information Processing Systems conference tutorial, a presentation,
  - http://www.cs.huji.ac.il/~nirf/Nips01-Tutorial/,
- Cooper, Herskovits: A Bayesian Method for the Induction of P.Networks from Data.
  - theory + K2 algorithm,
  - www.genetics.ucla.edu/labs/sabatti/Stat180/bayesNet.pdf,
- Heckerman: A Tutorial on Learning With Bayesian Networks.
  - a theoretical paper, "easy to read"
  - research.microsoft.com/apps/pubs/default.aspx?id=69588,
- Buntine: Operations for Learning with Graphical Models.
  - a general, complete and extensive description,
  - http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.52.696&rep=rep1&type=pdf.