Graphical probabilistic models - learning from data

## Jiří Kléma

Department of Cybernetics,
FEE, CTU at Prague

## IDA <br> Intelligent Data Analysis

Intelligent Data
RESEARCH GRoup

```
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?
- sample set $D$ contains $M$ samples = concurrent observations of all the variables,
- FAMILY example: $d_{m}=\left\{F O_{m}, B P_{m}, L O_{m}, D O_{m}, H B_{m}\right\}, m=1 \ldots M$,
- 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 $N(\{f o, b p, d o, l o, h b\}), \ldots, N(\{\neg f o, \neg b p, \neg d o, \neg l o, \neg h b\})$,
- representation close to the joint probability distribution.



## 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

$$
\begin{aligned}
L(\Theta: D) & =\prod_{m=1}^{M} \operatorname{Pr}\left(d_{m}: \Theta\right)=\prod_{m=1}^{M} \operatorname{Pr}\left(F O_{m}, B P_{m}, L O_{m}, D O_{m}, H B_{m}: \Theta\right)= \\
& =\prod_{m=1}^{M} \operatorname{Pr}\left(F O_{m}: \Theta\right) \operatorname{Pr}\left(B P_{m}: \Theta\right) \operatorname{Pr}\left(L O_{m} \mid F O_{m}: \Theta\right) \ldots \operatorname{Pr}\left(H B_{m} \mid D O_{m}: \Theta\right)
\end{aligned}
$$

- for general Bayesian network

$$
\begin{aligned}
L(\Theta: D) & =\prod_{m=1}^{M} \operatorname{Pr}\left(d_{m}: \Theta\right)=\prod_{m=1}^{M} \operatorname{Pr}\left(P_{1 m}, P_{2 m}, \ldots, P_{n m}: \Theta\right)= \\
& =\prod_{j=1}^{n} \prod_{m=1}^{M} \operatorname{Pr}\left(P_{j} \mid \operatorname{parents}\left(P_{j}\right): \Theta_{j}\right)=\prod_{j=1}^{n} L_{j}\left(\Theta_{j}: D\right)
\end{aligned}
$$

- under the assumption of independence of parameters, likelihood can be decomposed
- contribution of each network node $L_{j}\left(\Theta_{j}: D\right)$ is determined (maximized) independently.


## Learning Bayesian network parameters from data

- the optimization task: $\widehat{\Theta_{j}}=\arg \max L_{j}\left(\Theta_{j}: D\right)$ is solved for each node,
- let us demonstrate for FO node, where $\Theta_{F O}=\{\operatorname{Pr}(f o)\}$
- let $N(f o)$ be the number of samples, where $F O_{j}=T R U E$
- $L_{F O}$ is maximized by putting its first derivative equal to 0

$$
\begin{aligned}
& L_{F O}\left(\Theta_{F O}: D\right)=\prod_{m=1}^{M} \operatorname{Pr}\left(F O: \Theta_{F O}\right)=\operatorname{Pr}(f o)^{N(f o)}(1-\operatorname{Pr}(f o))^{M-N(f o)} \\
& \frac{\partial L_{F O}(\operatorname{Pr}(f o): D)}{\partial \operatorname{Pr}(f o)}=0 \rightarrow \operatorname{Pr}(f o)=\frac{N(f o)}{M}
\end{aligned}
$$

- the generalized formula for ML parameter estimation is intuitively obvious

$$
\widehat{\theta}_{P_{j} \mid \text { parents }\left(P_{j}\right)}=\frac{N\left(P_{j}, \operatorname{parents}\left(P_{j}\right)\right)}{N\left(\operatorname{parents}\left(P_{j}\right)\right)} \approx \operatorname{Pr}\left(P_{j} \mid \operatorname{parents}(P j)\right)
$$

- 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

$$
\operatorname{Pr}(\Theta \mid D)=\frac{\operatorname{Pr}(D \mid \Theta) \times \operatorname{Pr}(\Theta)}{\operatorname{Pr}(D)} \Leftrightarrow \text { posterior }=\frac{\text { likelihood } \times \text { prior }}{\text { prob of data }}
$$




- MAP estimate of parameters: $\widehat{\theta}_{P_{j} \mid \text { parents }\left(P_{j}\right)}=\frac{N\left(P_{j}, \text { parents }\left(P_{j}\right)\right)+\alpha-1}{N\left(\text { parents }\left(P_{j}\right)\right)+\alpha+\beta-2}$


## 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 $\operatorname{Pr}(f o)$ then $\operatorname{Pr}(d o \mid f o, b p)$ ? see graphs ...


## Parameter learning from data - complete observations

- What is the probability that family is out?
$-\operatorname{Pr}(f o)=?$
- all samples can be used...
$-\operatorname{Pr}(f o)=\frac{\sum_{m=1}^{M} \delta\left(F O^{m}, f o\right)}{M}$

- What is the dog out prob given $f o$ and $b p$ ?
$-\operatorname{Pr}(d o \mid f o, b p)=?$
- Condition is met only in $1.5 \%$ of samples.
$-\operatorname{Pr}(f o)=0.15, \operatorname{Pr}(b p)=0.01$,
- $F O$ and $B P$ independent variables.



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

- What is the probability that family is out?
$-\operatorname{Pr}(f o)=?$
- Incomplete data $=$ less information
- considerably longer computational time,
- the final estimate "a bit less exact only".
- What is the dog out prob given $f o$ and $b p$ ?
$-\operatorname{Pr}(d o \mid f o, b p)=?$
- Incomplete data $=$ less information
- comparison is inconclusive.




## 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 $\operatorname{Pr}\left(P_{j+1} \mid P_{1}, \ldots, P_{j}\right)=\operatorname{Pr}\left(P_{j+1} \mid \operatorname{parents}\left(P_{j+1}\right)\right), \operatorname{parents}\left(P_{j+1}\right) \subseteq\left\{P_{1}, \ldots, P_{j}\right\}$,

- the algorithm can be illustrated on a simple three variable example:

1. select a permutation $\pi: \pi\left(P_{1}\right)=1, \pi\left(P_{2}\right)=2$ a $\pi\left(P_{3}\right)=3$,
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 $\operatorname{Pr}\left(P_{j+1} \mid P_{1}, \ldots, P_{j}\right)=\operatorname{Pr}\left(P_{j+1} \mid \operatorname{parents}\left(P_{j+1}\right)\right), \operatorname{parents}\left(P_{j+1}\right) \subseteq\left\{P_{1}, \ldots, P_{j}\right\}$,

- the algorithm can be illustrated on a simple three variable example:

1. select a permutation $\pi: \pi\left(P_{1}\right)=1, \pi\left(P_{2}\right)=2$ a $\pi\left(P_{3}\right)=3$,
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 $\mathcal{O}\left(2^{n}\right)$ operations per single permutation,
* among others, the conditional probability $\operatorname{Pr}\left(P_{n} \mid P_{1}, \ldots, P_{n-1}\right)$ 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,

$$
\log L(G: D)=\log \prod_{m=1}^{M} \operatorname{Pr}\left(d_{m}: G\right)=M \sum_{j=1}^{n}\left(I\left(P_{j}: \operatorname{parents}\left(P_{j}\right)^{G}\right)-H\left(P_{j}\right)\right)
$$

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

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

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

$$
\operatorname{Pr}(D \mid G)=\int \operatorname{Pr}\left(D \mid G, \Theta_{G}\right) \times \operatorname{Pr}\left(\Theta_{G} \mid G\right) d \Theta
$$

- MLE concerns solely the best parametrization

$$
L(G: D)=\operatorname{Pr}\left(D \mid G, \widehat{\Theta_{G}}\right)
$$

## 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: $q_{i} \ldots$ the number of unique instantiations of $P_{i}$ parents,
$r_{i} \ldots$ the number of distinct $P_{i}$ values,
- then, a network has: $K=\sum_{i=1}^{n} q_{i}\left(r_{i}-1\right)$ independent parameters,

$$
B I C=-\frac{K}{2} \log _{2} M+\log _{2} L(G: D)=-\frac{K}{2} \log _{2} M-M \sum_{i=1}^{n} H\left(P_{i} \mid \operatorname{parents}\left(P_{i}\right)^{G}\right)
$$

- first addend: network complexity penalty (K $\uparrow$ BIC $\downarrow$ ),
- second addend: network likelihood
(mutual information between nodes and their parents $\uparrow H(\mid) \downarrow$ BIC $\uparrow$ ),
- how to enumerate conditional entropy?
* $N_{i j} \ldots$ the number of samples, where parents $\left(P_{i}\right)$ take the $j$-th instantiation of values,
* $N_{i j k} \ldots$ the number of samples, where $P_{i}$ takes the k-th value and parents $\left(P_{i}\right)$ the j-th instantiation of values,

$$
H\left(P_{i} \mid \operatorname{parents}\left(P_{i}\right)^{G}\right)=-\sum_{j=1}^{q_{i}} \sum_{k=1}^{r_{i}} \frac{N_{i j}}{M} \frac{N_{i j k}}{N_{i j}} \log _{2} \frac{N_{i j k}}{N_{i j}}=-\sum_{j=1}^{q_{i}} \sum_{k=1}^{r_{i}} \frac{N_{i j k}}{M} \log _{2} \frac{N_{i j k}}{N_{i j}}
$$

## 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
- random restarts, simulated annealing, TABU search.


## Structure learning - K2 algorithm

- Cooper and Herskovitz (1992), it approaches the naïve approach mentioned above,
- advantage
- complexity is $\mathcal{O}\left(m, u^{2}, n^{2}, r\right), u \leq n \rightarrow \mathcal{O}\left(m, n^{4}, r\right)$
$* m \ldots$ the number of samples, $n \ldots$ the number of variables,
* $r \ldots$ max number of distinct variable values, $u \ldots$ max number of parents,
- disadvantages
- topological sort of network variables $\pi$ must be given/known,
- greedy search results in locally optimal solution.
- it expresses the prob $\operatorname{Pr}(G, D)$ as the following function

$$
g\left(P_{i}, \text { parents }\left(P_{i}\right)\right)=\prod_{j=1}^{q_{i}} \frac{\left(r_{i}-1\right)!}{\left(N_{i j}+r_{i}-1\right)!} \prod_{k=1}^{r_{i}} N_{i j k}!
$$

- $q_{i} \ldots$ number of unique instantiations of parents $\left(P_{i}\right), r_{i} \ldots$ number of distinct $P_{i}$ values,
- $N_{i j} \ldots$ number of samples, where parents $\left(P_{i}\right)$ take $j$-th instantiation of values,
- $N_{i j k} \ldots$ number of samples, where $P_{i}$ takes k -th value and parents $\left(P_{i}\right)$ j-th instantiation of values,
- separable criterion - it can be computed node by node.


## Structure learning - K2 algorithm

- algorithm $\mathrm{K} 2(\pi, u, D)$ :
for $i=1: n \%$ follow the topological sort of variables $\pi$
parents $\left(P_{\pi_{i}}\right)=\emptyset \%$ in the beginning, the set of parents is always empty $G_{o l d}=\mathrm{g}\left(P_{\pi_{i}}\right.$, parents $\left.\left(P_{\pi_{i}}\right)\right) \%$ initialize the node value while $\mid$ parents $\left(P_{\pi_{i}}\right) \mid \leq u \%$ the number of parents must not exceed $u$
$j^{*}=\underset{j=1 \ldots i-1, P_{\pi_{j}} \notin \operatorname{parents}\left(P_{\pi_{i}}\right)}{\arg \max } g\left(P_{\pi_{i}}, \operatorname{parents}\left(P_{\pi_{i}}\right) \cup P_{\pi_{j}}\right)$
$\% P_{\pi_{j}^{*}}$ 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
$G_{n e w}=g\left(P_{\pi_{i}}\right.$, parents $\left.\left(P_{\pi_{i}}\right) \cup P_{\pi_{j}^{*}}\right)$
if $G_{\text {new }}>G_{\text {old }}$ then
$G_{\text {old }}=G_{\text {new }}$
parents $\left(P_{\pi_{i}}\right)=$ parents $\left(P_{\pi_{i}} \cup P_{\pi_{j}^{*}}\right)$
else
STOP \% the node value cannot be further improved, stop its processing


## K2 - locality of greedy search, illustration

- 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

- $\mathrm{MCMC}=$ 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 $P(G)$

- use samples, apply criteria such as BIC or Bayesian,

3. generate a "neighbor" $S$ of the given graph $G$

- insert/remove an edge, change edge direction,
- check the graph acyclicity constraint,
- prob of transition from $G$ to $S$ is function of $Q(G, S)$,

4. evaluate the neighbor graph $P(S)$,
5. accept or reject the transition to $S$

- generate $\alpha$ from $\mathrm{U}(0,1)$ (uniform distribution),

- if $\alpha<\frac{P(S) Q(G, S)}{P(G) Q(S, G)}$ then accept the transition $G \rightarrow S$,

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.

- $G_{1}$ gradually evaluated by three criteria:
- likelihood: ML parameters first $\operatorname{Pr}\left(p_{1}\right)=\operatorname{Pr}\left(p_{2}\right)=\frac{9}{16}, \operatorname{Pr}\left(p_{3}\right)=\frac{1}{8}$

$$
\begin{aligned}
\ln L\left(G_{1}: D\right) & =\sum_{m=1}^{16} \operatorname{Pr}\left(d_{m}: G_{1}\right)= \\
& =2 \ln \left(\frac{7}{16} \frac{9}{16} \frac{1}{8}\right)+3 \ln \left(\frac{9}{16} \frac{9}{16} \frac{7}{8}\right)+10 \ln \left(\frac{9}{16} \frac{7}{16} \frac{7}{8}\right)+\ln \left(\frac{7}{16} \frac{7}{16} \frac{7}{8}\right)=-27.96
\end{aligned}
$$

## Structure learning－3DAG example

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

$$
\begin{aligned}
\ln L\left(G_{1}: D\right) & =-M \sum_{i=1}^{3} H\left(P_{i} \mid \operatorname{parents}\left(P_{i}\right)^{G_{1}}\right)= \\
& =-16\left[-2\left(\frac{9}{16} \ln \frac{9}{16}+\frac{7}{16} \ln \frac{7}{16}\right)-\left(\frac{1}{8} \ln \frac{1}{8}+\frac{7}{8} \ln \frac{7}{8}\right)\right]=-27.96
\end{aligned}
$$

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

$$
B I C\left(G_{1}: D\right)=-\frac{K}{2} \ln M+\ln L\left(G_{1}: D\right)=-\frac{3}{2} \ln 16-27.96=-32.12
$$

－Bayesian score

$$
\begin{aligned}
\ln \operatorname{Pr}\left(D \mid G_{1}\right) & =\ln \prod_{i=1}^{3} g\left(P_{i}, \text { parents }\left(P_{i}\right)^{G_{1}}\right)=\sum_{i=1}^{3} \sum_{j=1}^{q_{i}} \sum_{k=1}^{r_{i}} \ln \frac{\left(r_{i}-1\right)!}{\left(N_{i j}+r_{i}-1\right)!} N_{i j k}!= \\
& =2(-\ln 17!+\ln 9!+\ln 7!)-\ln 17!+\ln 2!+\ln 14!=-31.98
\end{aligned}
$$

Natural logarithm is applied to match Matlab BN Toolbox．
Logarithm base change does not change ordering of model evaluations．

## Structure learning - 3DAG example

$$
\begin{array}{|c|c|}
\hline \mathrm{G}_{5} & \operatorname{Pr}\left(\mathrm{p}_{1}\right)=.5 \\
\underset{\mathrm{P}_{1}}{\sim} & \operatorname{Pr}\left(\mathrm{p}_{3} \mid \mathrm{p}_{2}\right)=.2 \\
\mathrm{P}_{2} & \operatorname{Pr}\left(\mathrm{p}_{3} \mid \neg \mathrm{p}_{2}\right)=.1 \\
\operatorname{Pr}\left(\mathrm{p}_{2} \mid \mathrm{p}_{1}\right)=.25 \\
\operatorname{Pr}\left(\mathrm{p}_{2} \mid \neg \mathrm{p}_{1}\right)=.75 \\
\hline
\end{array}
$$

| $\mathrm{G}_{1}$ | $\mathrm{G}_{2}$ | $\mathrm{G}_{3}$ | $\mathrm{G}_{4}$ | $\mathrm{G}_{5}$ |
| :--- | :--- | :--- | :--- | :--- |
| -27.96 | -25.59 | -26.12 | -26.70 | -24.33 |
| -31.98 | -30.56 | -31.78 | -32.32 | -30.92 |
| -32.12 | -31.14 | -31.67 | -32.25 | -31.26 |
| $\mathrm{G}_{6}$ | $\mathrm{G}_{7}$ | $\mathrm{G}_{8}$ | $\mathrm{G}_{9}$ | $\mathrm{G}_{10}$ |
| -25.32 | -23.75 | -24.64 | -24.86 | -25.75 |
| -31.03 | -30.36 | -30.56 | -31.33 | -33.04 |
| -33.64 | -30.68 | -32.96 | -31.79 | -34.07 |
| $\mathbf{G}_{11}$ |  |  |  |  |
| $-\mathbf{2 3 . 3 8}$ | MLE | Bayesian | BIC |  |
| -31.62 | MLE |  |  |  |

- 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.


## Recommended reading, lecture resources

- 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.

