

B-Course is a web-based data analysis tool for Bayesian modeling, in particular dependence classification modeling. It can also be used as an interactive tutorial which provides you with data sets that have been prepared in advance. B-Course can be used as an educational tool for an introductory course in Bayesian modeling, or as a reference tool for those interested in the field. B-Course can be used for educational purposes, but you must not use it for commercial purposes. If you are interested in using B-Course for educational purposes, please contact the authors.

New features added to the version of B-Course (2.0), including a new classification tree and a new reference software. You are welcome to try them out. Thank you for your feedback.

## Welcome to B-Course

# Learning Bayesian Networks

### The Basics of Modeling

B-Course Service is hosted by Complex Systems Computation Group, CSCo, Helsinki Institute for Information Technology

Along the way, you will find references to B-Course library, where you can find information on how to use B-Course to study the text. If you are interested in truly understanding what is going on in the analysis, you can also find information on how to use B-Course as a software tool to help you in the analysis of your data. If you publish the results, we as the designers of B-Course would appreciate that you acknowledge that the results were obtained by using B-Course. A good reference to B-Course is:

P. Myllymäki, T. Silander, H. Tirri, P. Uronen, B-Course: A Web-Based Tool for Bayesian and Causal Data Analysis. *International Journal on Artificial Intelligence Tools*, Vol 11, No. 3 (2002) 369-387.

# Aspects in learning

- Learning the parameters of a Bayesian network
  - Marginalizing over all all parameters
  - Equivalent to choosing the expected parameters
- Learning the structure of a Bayesian network
  - Marginalizing over the structures not computationally feasible
  - Model selection

# A Bayesian network

$P(\text{Cloudy})$

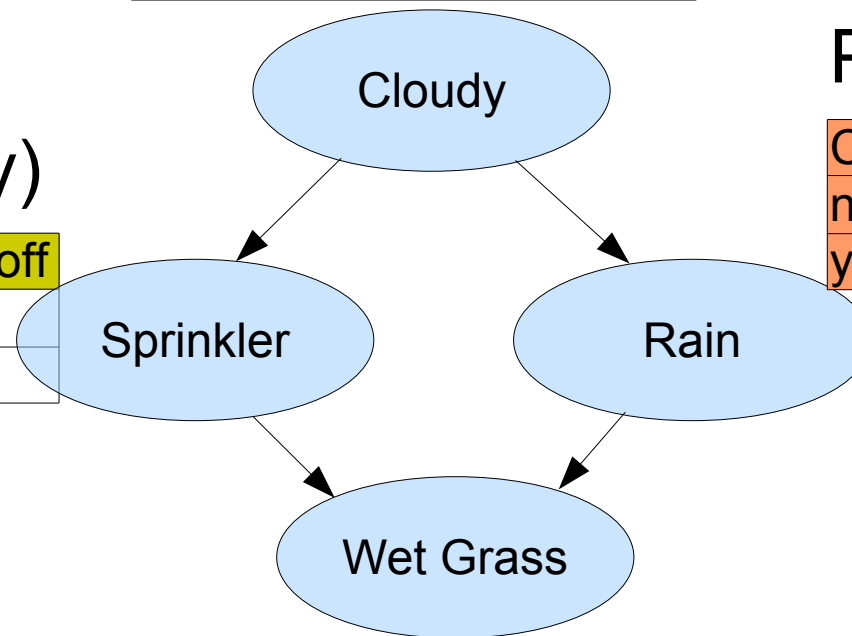
	Cloudy=no	Cloudy=yes
	0.5	0.5

$P(\text{Rain} \mid \text{Cloudy})$

Cloudy	Rain=yes	Rain=no
no	0.2	0.8
yes	0.8	0.2

$P(\text{Sprinkler} \mid \text{Cloudy})$

Cloudy	Sprinkler=on	Sprinkler=off
no	0.5	0.5
yes	0.9	0.1



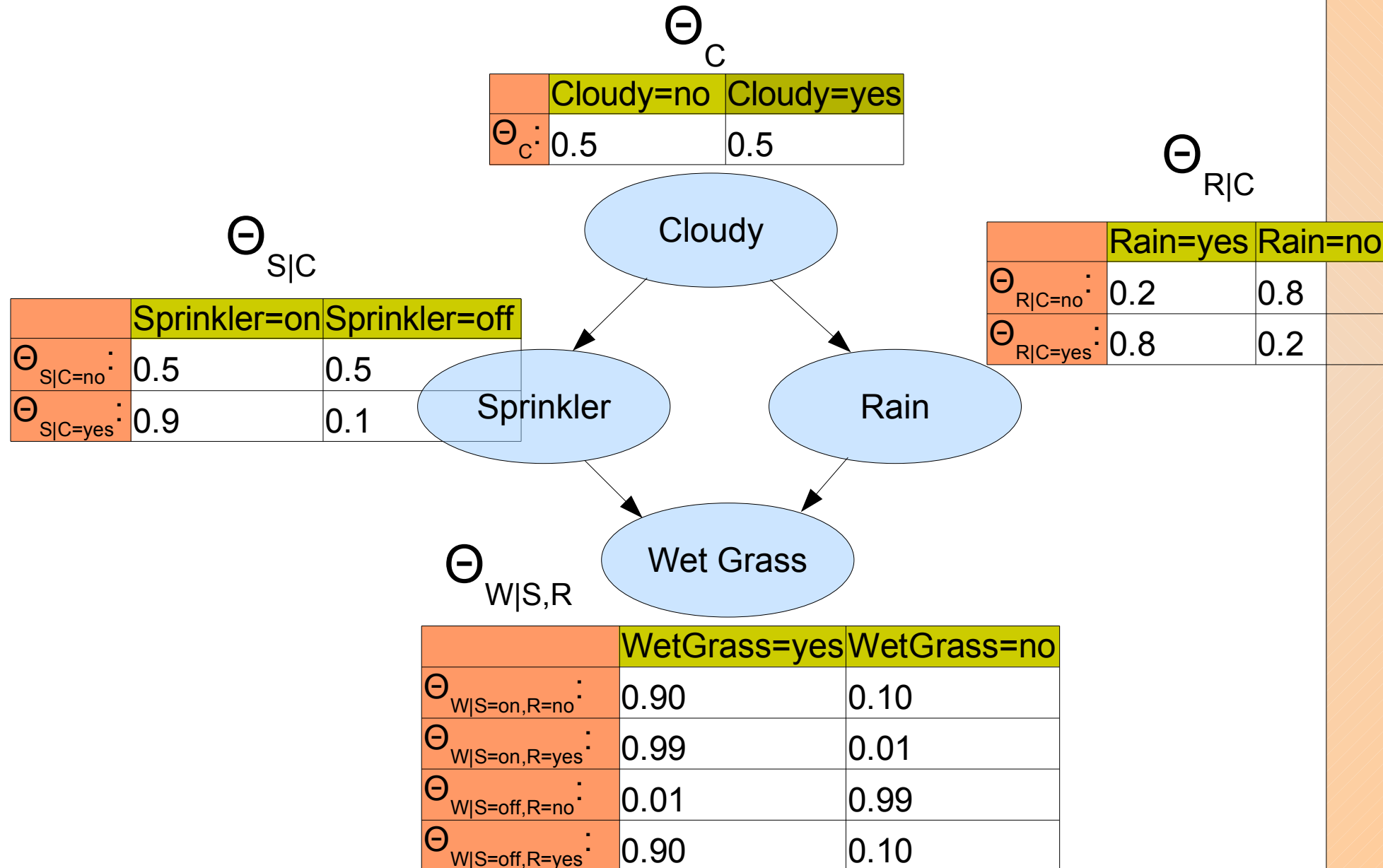
$P(\text{WetGrass} \mid \text{Sprinkler}, \text{Rain})$

Sprinkler	Rain	WetGrass=yes	WetGrass=no
on	no	0.90	0.10
on	yes	0.99	0.01
off	no	0.01	0.99
off	yes	0.90	0.10

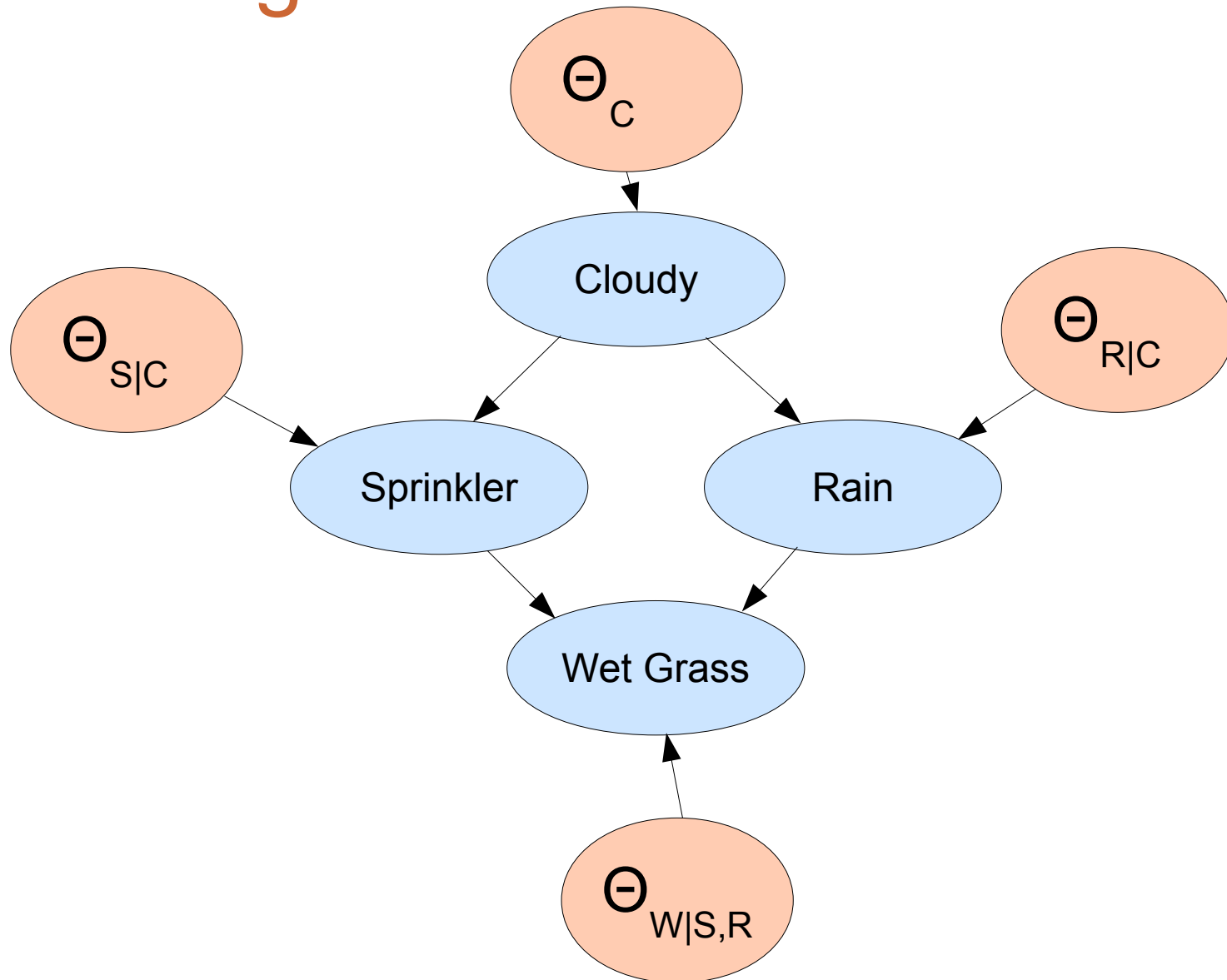
# Learning the parameters

- Given the data  $D$ , how should I fill the conditional probability tables?
- Bayesian answer:
  - You should not. If you do not know them, you will have a priori and a posteriori distributions for them.
  - They are many, but again, the independence comes to rescue.
  - Once you have distribution of parameters, you can do the prediction by model averaging.
  - Very similar to Bernoulli case.

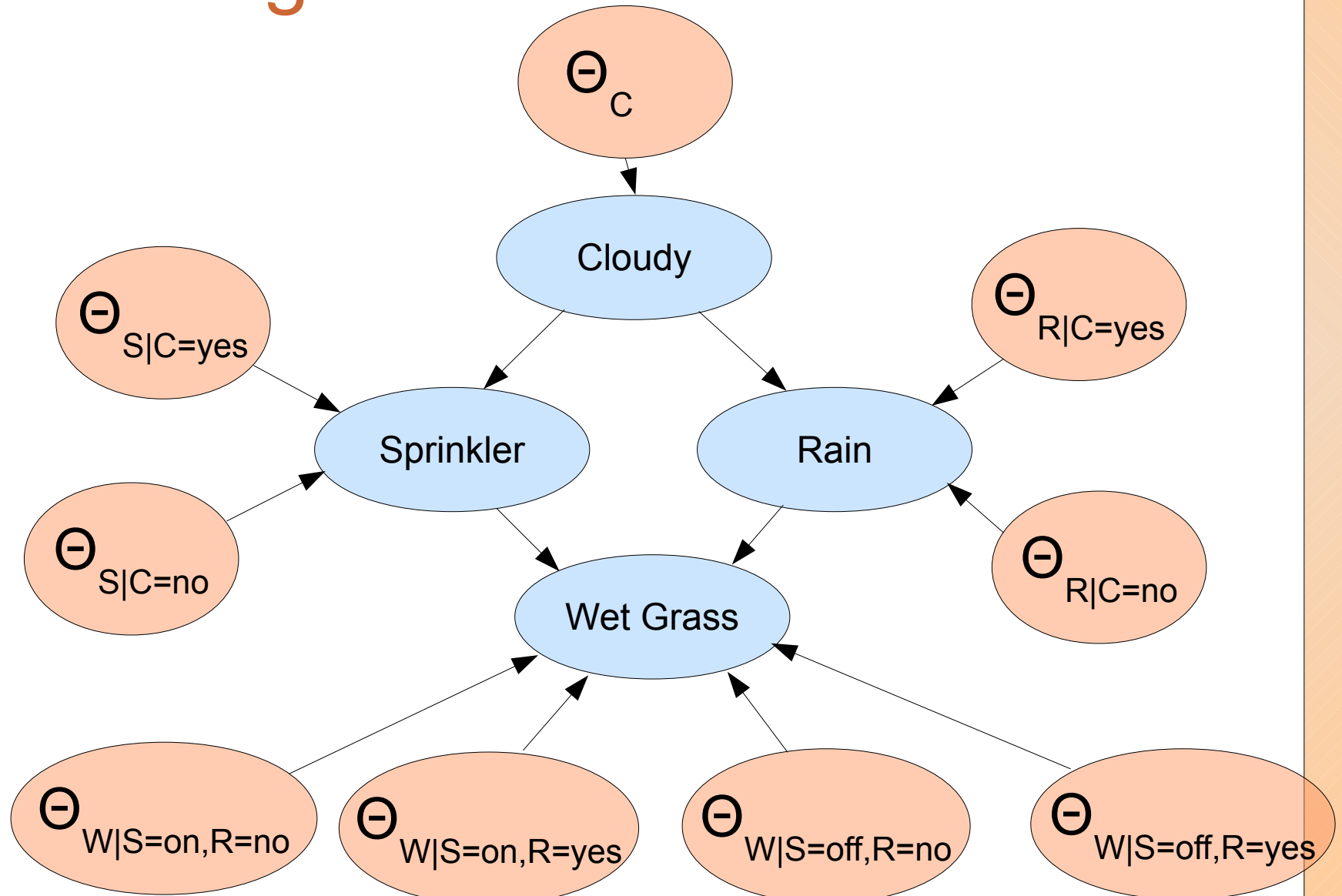
# A Bayesian network revisited



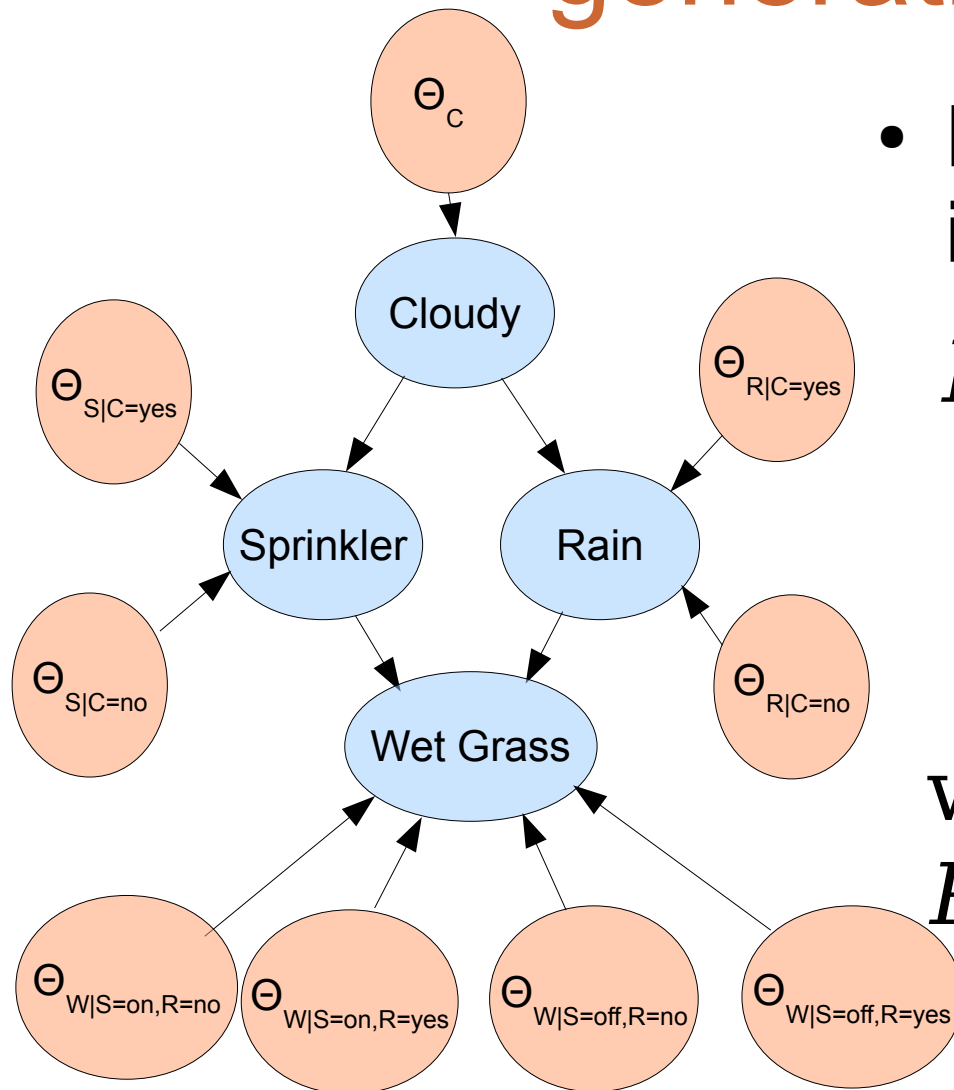
# A Bayesian network as a generative model



# A Bayesian network as a generative model



# A Bayesian network as a generative model



- Parameters are independent a priori:

$$P(\Theta) = \prod_{i=1}^n P(\Theta_i)$$

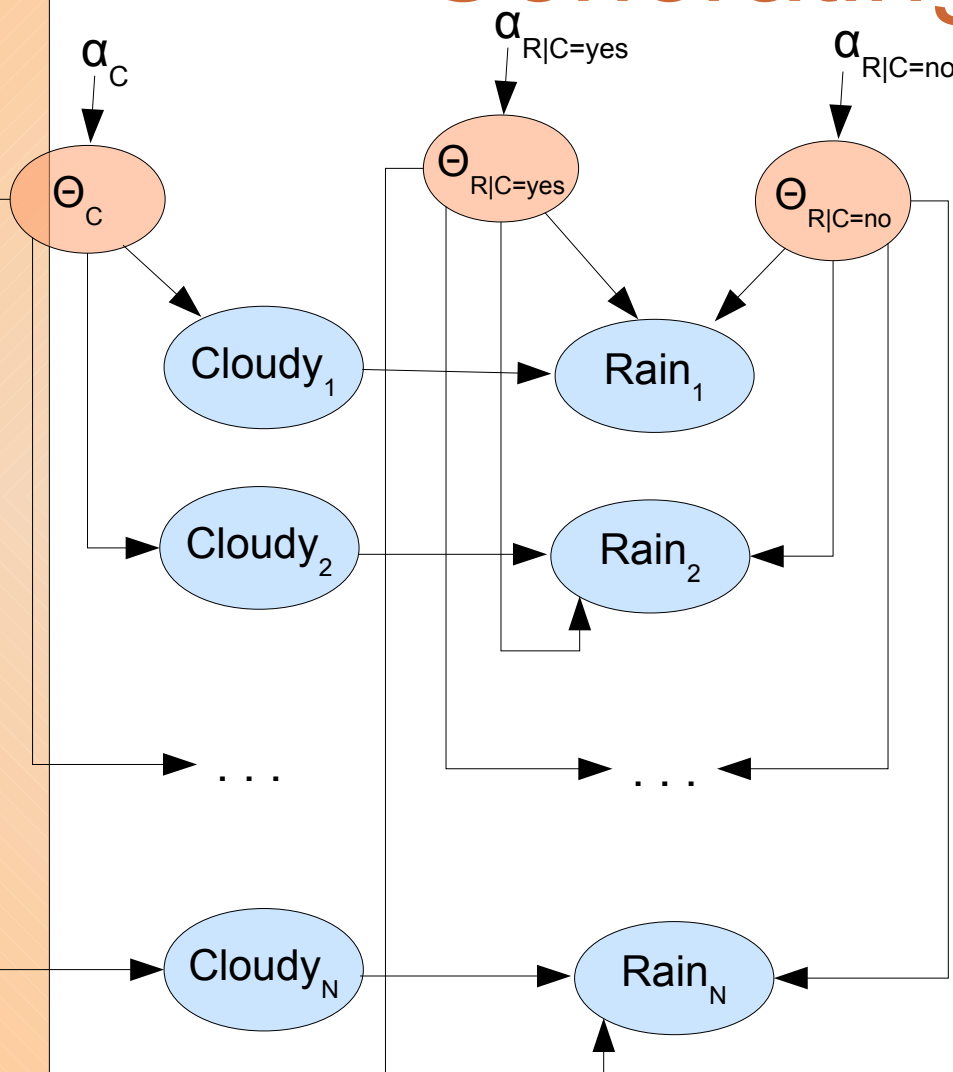
$$= \prod_{i=1}^n \prod_{j=1}^{q_i} P(\Theta_{i|j}),$$

where

$$P(\Theta_{i|j}) = \text{Dir}(\alpha_1, \dots, \alpha_{r_i}).$$



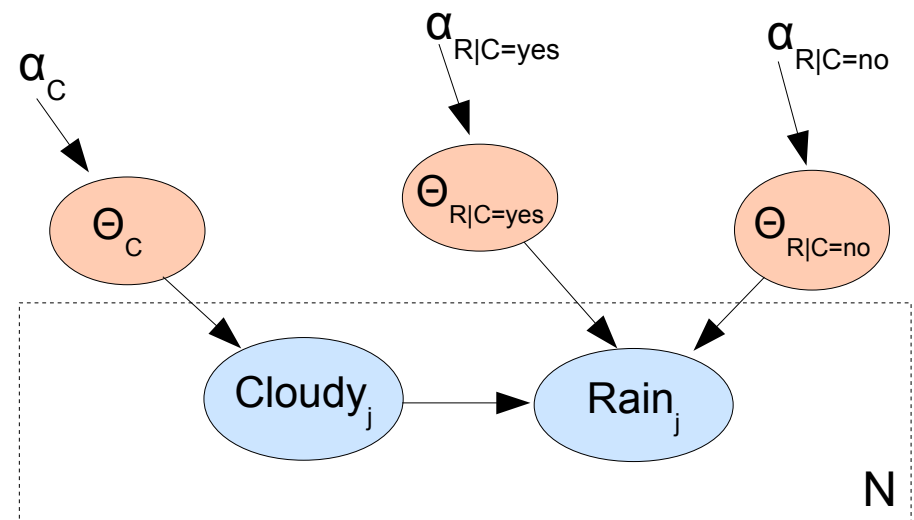
# Generating a data set



i.i.d, isn't it

D	Cloudy	Rain
$d_1$	yes	no
$d_2$	no	yes
...	...	...
$d_N$	no	yes

- Plate notation:



# Likelihood $P(D|\Theta, G)$

- For one data vector it was:

$$P(x_1, x_2, \dots, x_n | G) = \prod_{i=1}^n P(x_i | pa_G(x_i)), \text{ or}$$

$$P(d_1 | G, \theta) = \prod_{i=1}^n \theta_{d_{1i} | pa_{1i}}, \text{ where } d_{1i} \text{ and } pa_{1i} \text{ are the}$$

value and the parent configuration of the variable  $i$  in data vector  $d_1$ .

$$P(d_1, d_2, \dots, d_N | G, \theta) = \prod_{j=1}^N \prod_{i=1}^n \theta_{d_{ji} | pa_{ji}} = \prod_{i=1}^n \prod_{k=1}^{r_i} \prod_{j=1}^{q_i} \theta_{ik|j}^{N_{ijk}},$$

where  $N_{ijk}$  is the number of data vectors with parent configuration  $j$  when variable  $i$  has the value  $k$ ,  $r_i$  and  $q_i$  are the numbers of values and parent configurations of the variable  $i$ .

# Bayesian network learning

$$N_C(q_C=1, r_C=2)$$

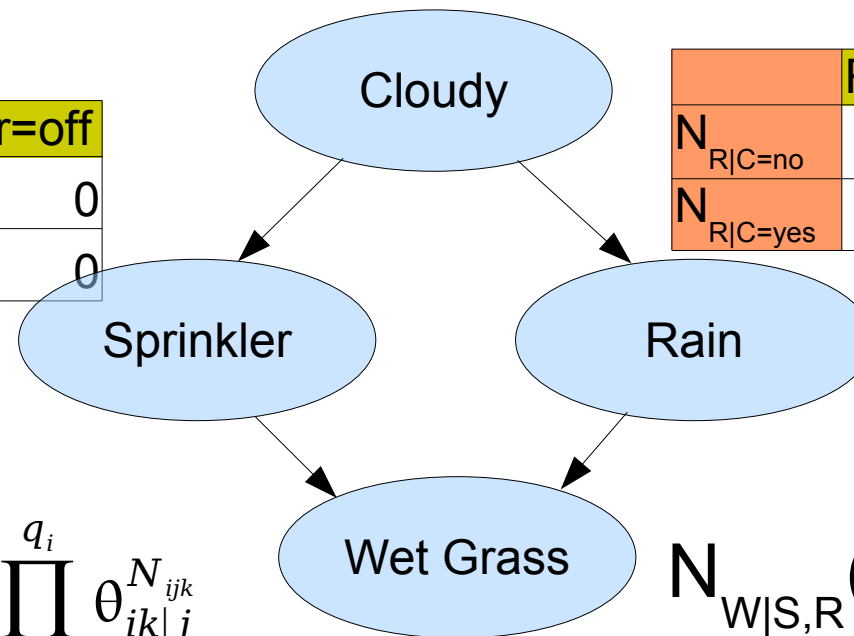
	Cloudy=no	Cloudy=yes
$N_C$	0	0

$$N_{R|C}(q_R=2, r_R=2)$$

	Rain=yes	Rain=no
$N_{R C=no}$	0	0
$N_{R C=yes}$	0	0

$$N_{S|C}(q_S=2, r_S=2)$$

	Sprinkler=on	Sprinkler=off
$N_{S C=no}$	0	0
$N_{S C=yes}$	0	0



$$N_{W|S,R}(q_W=4, r_W=2)$$

	WetGrass=yes	WetGrass=no
$N_{W S=on,R=no}$	0	0
$N_{W S=on,R=yes}$	0	0
$N_{W S=off,R=no}$	0	0
$N_{W S=off,R=yes}$	0	0

$$P(D|G, \theta) = \prod_{i=1}^n \prod_{k=1}^{r_i} \prod_{j=1}^{q_i} \theta_{ik|j}^{N_{ijk}}$$

- $i$  picks the variable (table)
- $j$  picks the row
- $k$  picks the column

# Bayesian network learning after (C,S,R,W)=[(no, on, yes, yes), (no,on,no,no)]

$$N_C(q_C=1, r_C=2)$$

	Cloudy=no	Cloudy=yes
$N_C$	1+1=2	0

$$N_{R|C}(q_R=2, r_R=2)$$

	Rain=yes	Rain=no
$N_{R C}$	1	1
$N_{R C}$	0	0

$$N_{S|C}(q_S=2, r_S=2)$$

	Sprinkler=on	Sprinkler=off
$N_{S C=no}$	1+1=2	0
$N_{S C=yes}$	0	0

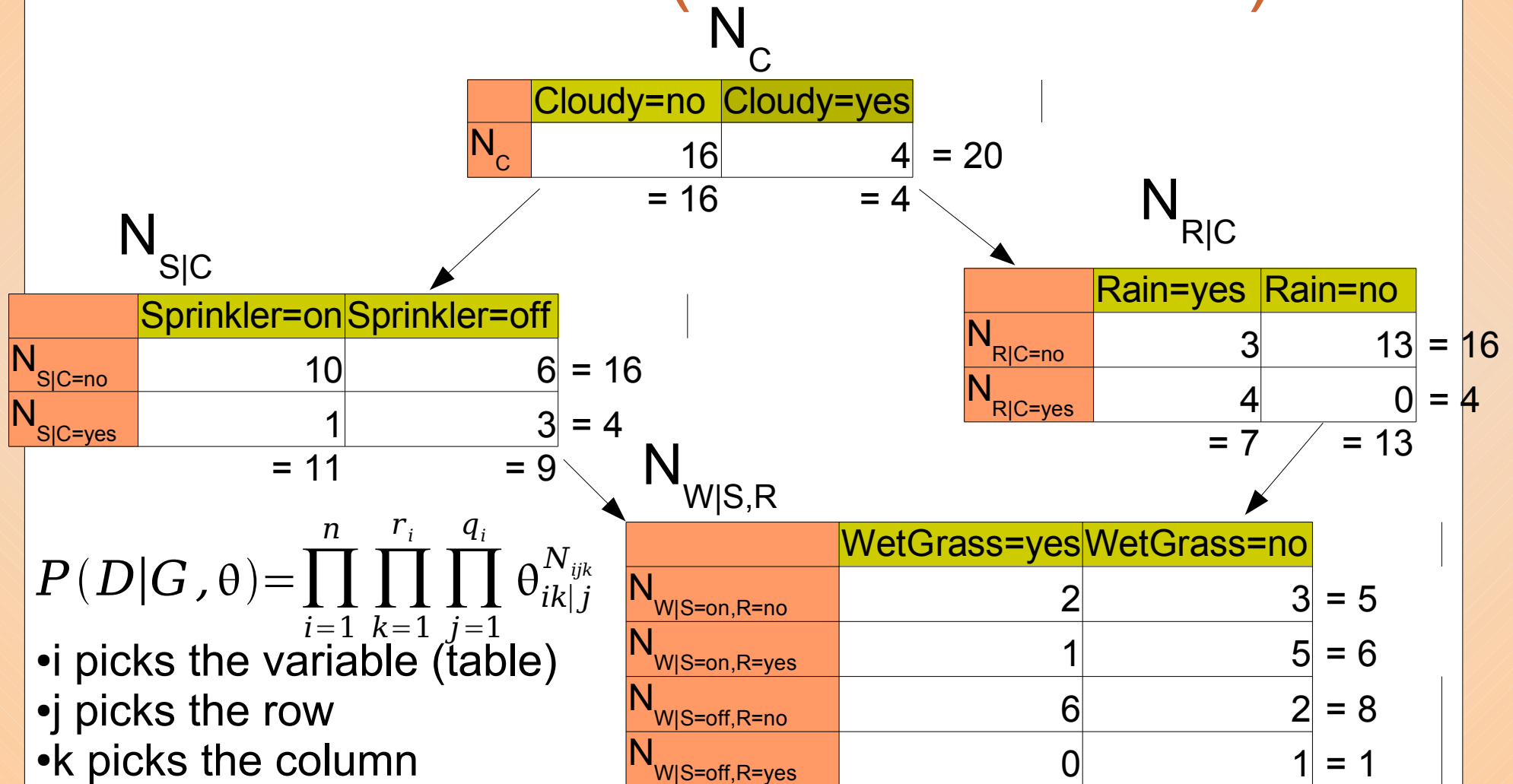
$$N_{W|S,R}(q_W=4, r_W=2)$$

	WetGrass=yes	WetGrass=no
$N_{W S=on,R=no}$	0	1
$N_{W S=on,R=yes}$	1	0
$N_{W S=off,R=no}$	0	0
$N_{W S=off,R=yes}$	0	0

$$P(D|G, \theta) = \prod_{i=1}^n \prod_{k=1}^{r_i} \prod_{j=1}^{q_i} \theta_{ik|j}^{N_{ijk}}$$

- $i$  picks the variable (table)
- $j$  picks the row
- $k$  picks the column
- $r_i$ , number of columns in table  $i$
- $q_i$ , number of rows in table  $i$

# Bayesian network learning after a while (20 data vectors)



# Maximum likelihood

- Since the parameters occur separately in likelihood we can maximize the terms independently:

$$P(D|G, \theta) = \prod_{i=1}^n \prod_{k=1}^{r_i} \prod_{j=1}^{q_i} \theta_{ijk}^{N_{ijk}} \quad \Rightarrow \quad \hat{\theta}_{ijk} = \frac{N_{ijk}}{\sum_{k'=1}^{r_i} N_{ijk'}}$$

- So you simply normalize the rows in the sufficient statistics tables to get ML-parameters.
- But these parameters may have zero probabilities:
  - not good for prediction; hear the Bayes call ....

# Learning the parameters - again

- Given the data  $D$ , how should I fill the conditional probability tables?
- Bayesian answer:
  - You should not. If you do not know them, you will have a priori and a posteriori distributions for them.
  - They are many, but again, the independence comes to rescue.
  - Once you have distribution of parameters, you can do the prediction by model averaging.
  - Very similar to the Bernoulli case.

# Prior x Likelihood

- A priori parameters independently Dirichlet:

$$P(\Theta|\alpha) = \prod_{i=1}^n P(\Theta_i) = \prod_{i=1}^n \prod_{j=1}^{q_i} P(\Theta_{i|j}) = \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{\Gamma(\sum_{k=1}^{r_i} \alpha_{ijk})}{\prod_{k=1}^{r_i} \Gamma(\alpha_{ijk})} \prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}-1}$$

- Likelihood compatible with conjugate prior:

$$P(D|G, \theta) = \prod_{i=1}^n \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}}$$

- Yields a simple posterior

$$P(\Theta|D, \alpha) = \prod_{i=1}^n \prod_{j=1}^{q_i} P(\Theta_{ij}|N_{ij}, \alpha_{ij}),$$

where  $P(\Theta_{ij}|N_{ij}, \alpha_{ij}) = \text{Dir}(N_{ij} + \alpha_{ij})$



# Predictive distribution $P(d|D, \alpha, G)$

- Posterior: 
$$P(\Theta|D, \alpha) = \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{\Gamma(\sum_{k=1}^{r_i} N_{ijk} + \alpha_{ijk})}{\prod_{k=1}^{r_i} \Gamma(N_{ijk} + \alpha_{ijk})} \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk} + \alpha_{ijk} - 1}$$

- Predictive distribution:

$$P(d|D, \alpha, G) = \int_{\theta} P(d, \theta|D, \alpha) d\theta = \int_{\theta} P(d|\theta) P(\theta|D, \alpha) d\theta$$

$$= \int_{\theta} \prod_{i=1}^n P(d_i|\theta_i) P(\theta_i|D, \alpha) d\theta$$

$$= \prod_{i=1}^n \int_{\theta_{ipa_i d_i}} \theta_{ipa_i d_i} P(\theta_{ipa_i d_i} | N_{ipa_i d_i}, \alpha_{ipa_i d_i}) d\theta_{ipa_i d_i}$$

$$= \prod_{i=1}^n \bar{\theta}_{ipa_i d_i} = \prod_{i=1}^n \frac{N_{ipa_i d_i} + \alpha_{ipa_i d_i}}{\sum_{k=1}^{r_i} N_{ipa_i k} + \alpha_{ipa_i k}}$$

# Predictive distribution

- This means that predictive distribution

$$P(d|D, \alpha, G) = \prod_{i=1}^n \frac{N_{ipa_i d_i} + \alpha_{ipa_i d_i}}{\sum_{k=1}^{r_i} N_{ipa_i k} + \alpha_{ipa_i k}}$$

can be achieved by just setting

$$\theta_{ijk} = \frac{N_{ijk} + \alpha_{ijk}}{N_{ij} + \alpha_{ij}}$$

- So just gather counts  $N_{ijk}$ , add  $\alpha_{ijk}$  to them and normalize.

# Being uncertain about the Bayesian network structure

- Bayesian says again:
  - If you do not know it, you should have an a priori and the a posteriori distribution for it.

$$P(G|D) = \frac{P(D|G)P(G)}{P(D)}$$

- Likelihood  $P(D|G)$  is called the *marginal likelihood* and with certain assumptions, it can be computed in closed form
- Normalizer we can just ignore.

# Prediction over model structures

$$\begin{aligned}
 P(X|D) &= \sum_M P(X|M, D) P(M|D) \\
 &= \sum_M \int_{\Theta} P(X|\Theta, M, D) P(\Theta|M, D) d\Theta P(M|D) \\
 &\propto \sum_M P(X|\bar{\Theta}(D), M) P(D|M) P(M) \\
 &= \sum_M P(X|\bar{\Theta}(D), M) \int_{\Theta} P(D|\Theta, M) P(\Theta|M) d\Theta P(M)
 \end{aligned}$$

- This summation is not feasible as it goes over a super-exponential number of model structures
- Does NOT reduce to using a single expected model structure, like what happens with the parameters
- Typically use only one (or a few) models with high posterior probability  $P(M | D)$

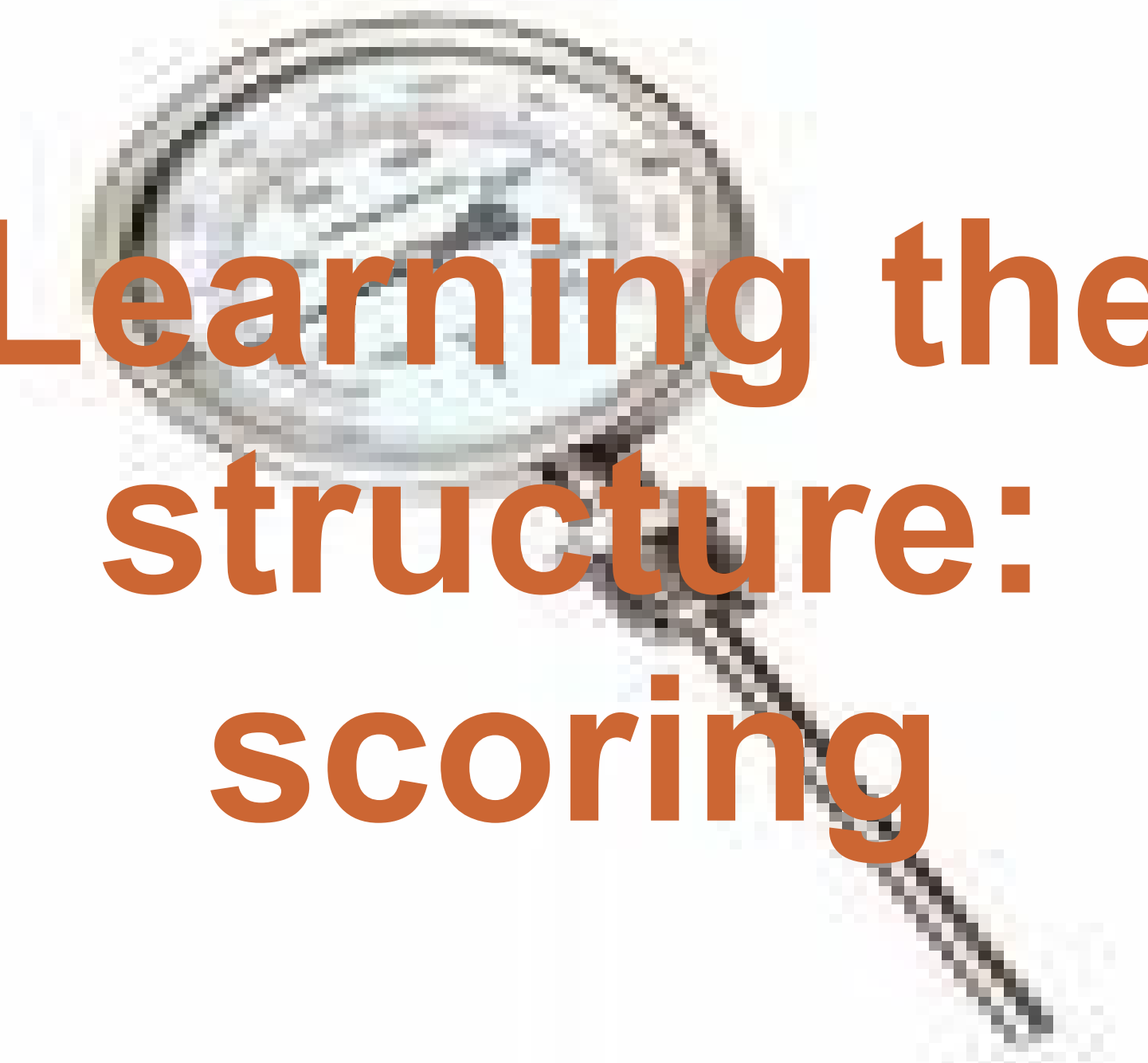
# Averaging over an equivalence class

- Boils down to using a single model (assuming uniform prior over the models within the equivalence class):

$$\begin{aligned} P(X|E) &= \sum_{M \in E} P(X|M, E) P(M|E) \\ &= |E| P(X|M) \frac{1}{|E|} \\ &= P(X|M) \end{aligned}$$

# Model Selection

- Problem: The number of possible structures for a given domain is more than exponential in the number of variables
- Solution: Use only one or a handful of "good" models
- Necessary components:
  - Scoring method (what is "good"?)
  - Search method (how to find good models?)



# Learning the structure: scoring

# Good models?

- In marginalization/summation/model averaging over all the model structures, the predictions are weighted by  $P(M | D)$ , the posteriors of the models given the data
- If have to select one (a few) model(s), it sounds reasonable to use model(s) with the largest weight(s)
- $P(M | D) = P(D | M)P(M)/P(D)$
- Relevant components:
  - The structure prior  $P(M)$
  - The marginal likelihood (the "evidence")  $P(D | M)$



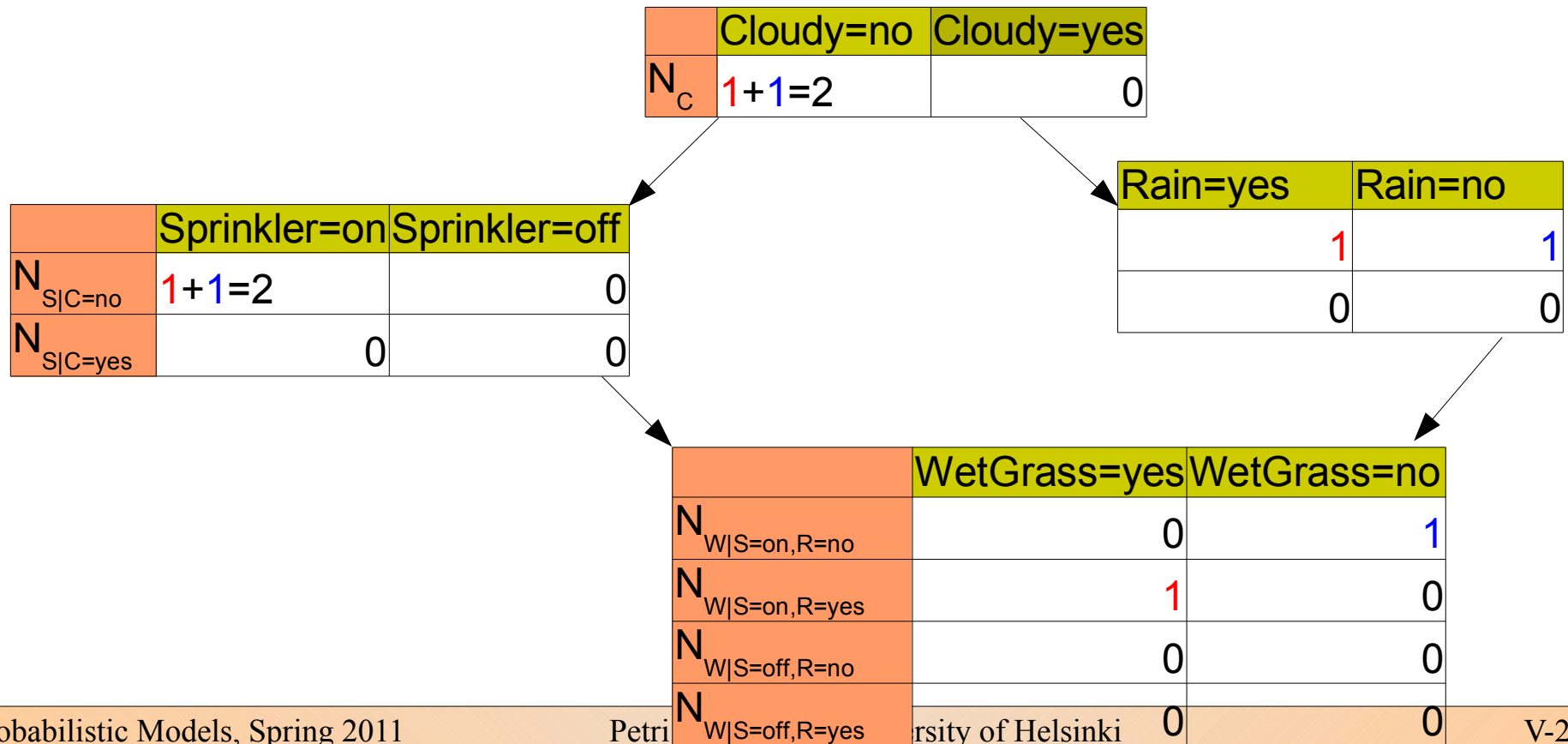
# How to set the structure prior $P(M)$ ?

- The "standard" solution: use the uniform prior (i.e., ignore the structure prior)
- Sometimes suggested:  $P(M)$  proportional to the number of arcs so that simple models more probable
  - Justification???
- Uniform over the equivalence classes?  
Proportional to the size of the equivalence class?  
What about the nestedness (full networks "contain" all the other networks)...?
- ...still very much an open issue

# Marginal likelihood $P(D|G, \alpha)$

$$P(D|G, \alpha) = P(d_1|G, \alpha) P(d_2|d_1, G, \alpha) \dots P(d_N|d_1, \dots, d_{N-1}, G, \alpha)$$

$$= \prod_{i=1}^n \prod_{j=1}^{q_i} \frac{\Gamma(\alpha_{ij})}{\Gamma(N_{ij} + \alpha_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N_{ijk} + \alpha_{ijk})}{\Gamma(\alpha_{ijk})}$$



# Computing the marginal likelihood

- Two choices:
  - 1 Calculate the sufficient statistics  $N_{ijk}$  and compute  $P(D | M)$  directly using the (gamma) formula on the previous slide
  - 2 Use the chain rule, and compute  $P(d_1, \dots, d_n | M) = P(d_1 | M)P(d_2 | d_1, M) \dots P(d_n | d_1, \dots, d_{n-1} | M)$  by using iteratively the predictive distribution (slide 18)
- OBS! The latter can be done in any order, and the result will be the same (remember Exercise 2?)!

# How to set the hyperparameters $\alpha$ ?

- Assuming...
  - a multinomial sample,
  - independent parameters,
  - modular parameters,
  - complete data,
  - likelihood equivalence,

...implies a certain parameter prior: BDe  
("Bayesian Dirichlet with likelihood equivalence")

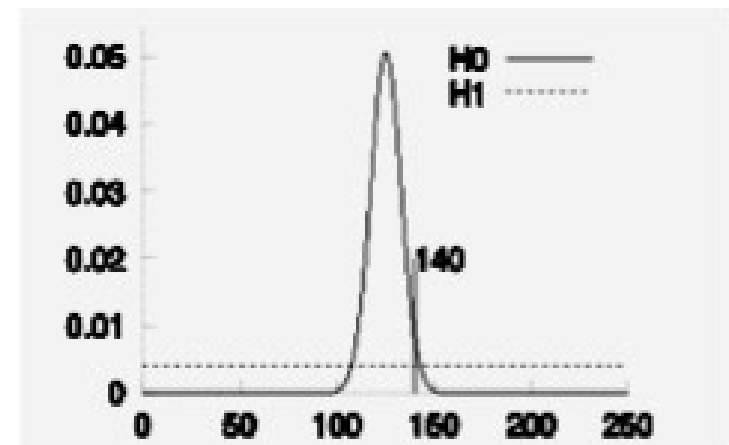
# BDeu

- Likelihood equivalence: two Markov equivalent model structures produce to the same predictive distribution
- Means also that  $P(D | M) = P(D | M')$  if  $M$  and  $M'$  equivalent
- Let  $\alpha_i = \sum_j \alpha_{ij}$ , where  $\alpha_{ij} = \sum_k \alpha_{ijk}$
- BDe means that  $\alpha_i = \alpha$  for all  $i$ , and  $\alpha$  is the **equivalent sample size**
- An important special case: BDeu ("u" for "uniform"):  $\alpha_{ijk} = \frac{\alpha}{q_i r_i}$ ,  $\alpha_{ij} = \frac{\alpha}{q_i}$

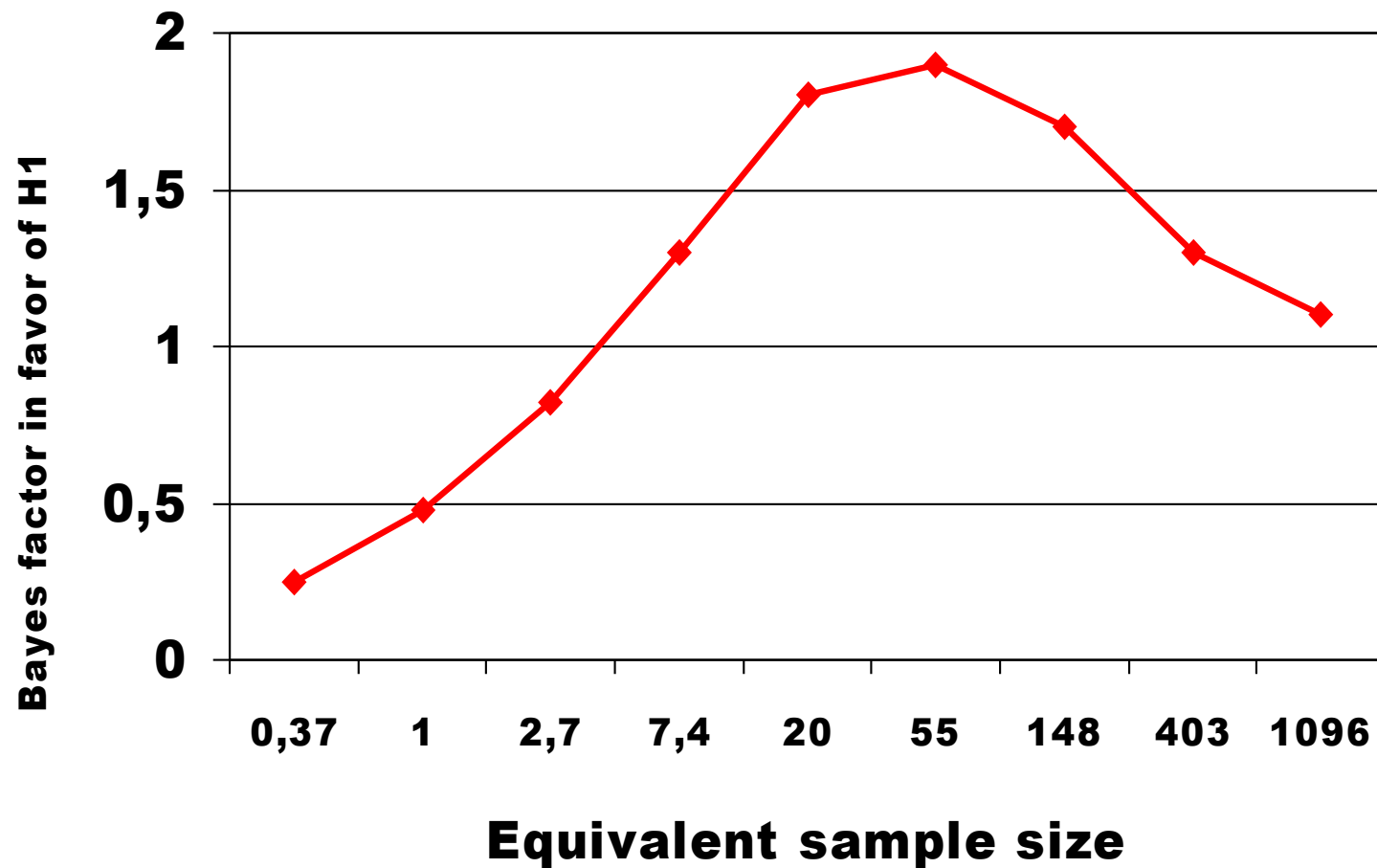
# Model selection in the Bernoulli case

- Toss a coin 250 times, observe  $D$ : 140 heads and 110 tails.
- Hypothesis  $H_0$ : the coin is fair ( $P(\Theta=0.5) = 1$ )
- Hypothesis  $H_1$ : the coin is biased
- Statistics:
  - The P-value is 7%
  - “suspicious”, but not enough for rejecting the null hypothesis (Dr. Barry Blight, The Guardian, January 4, 2002)
- Bayes:
  - Let’s assume a prior, e.g. Beta( $a,a$ )
  - Compute the Bayes factor

$$\frac{P(D|H_1)}{P(D|H_0)} = \frac{\int P(D|\theta, H_1, a) P(\theta|H_1, a) d\theta}{1/2^{250}}$$



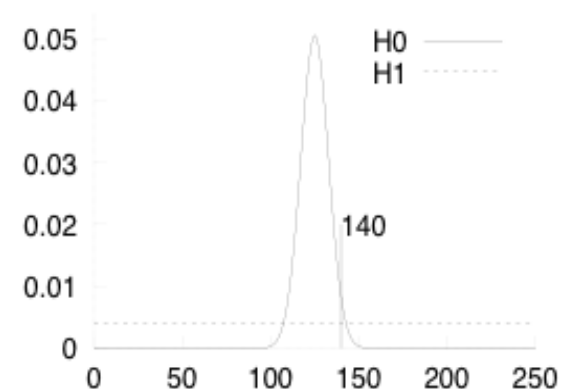
# Equivalent sample size and the Bayes Factor



# A slightly modified example

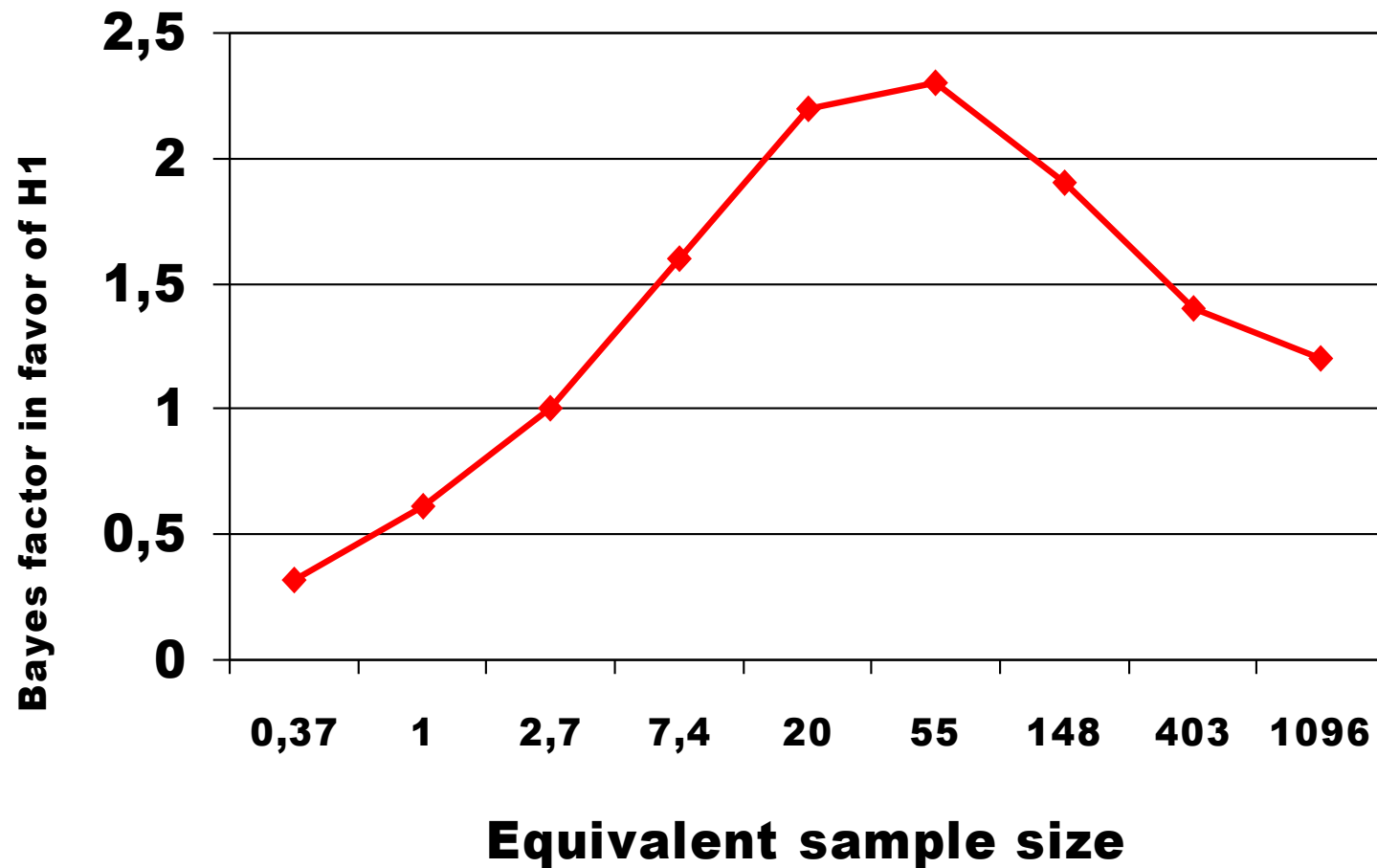
- Toss a coin 250 times, observe  $D = 141$  heads and 109 tails.
- Hypothesis  $H_0$ : the coin is fair ( $P(\Theta=0.5) = 1$ )
- Hypothesis  $H_1$ : the coin is biased
- Statistics:
  - The P-value is 4,97%
  - *Reject the null hypothesis at a significance level of 5%*
- Bayes:
  - Let's assume a prior, e.g. Beta(a,a)
  - Compute the Bayes factor

$$\frac{P(D|H_1)}{P(D|H_0)} = \frac{\int P(D|\theta, H_1, a) P(\theta|H_1, a) d\theta}{1/2^{250}}$$





# Equivalent sample size and the Bayes Factor (modified example)



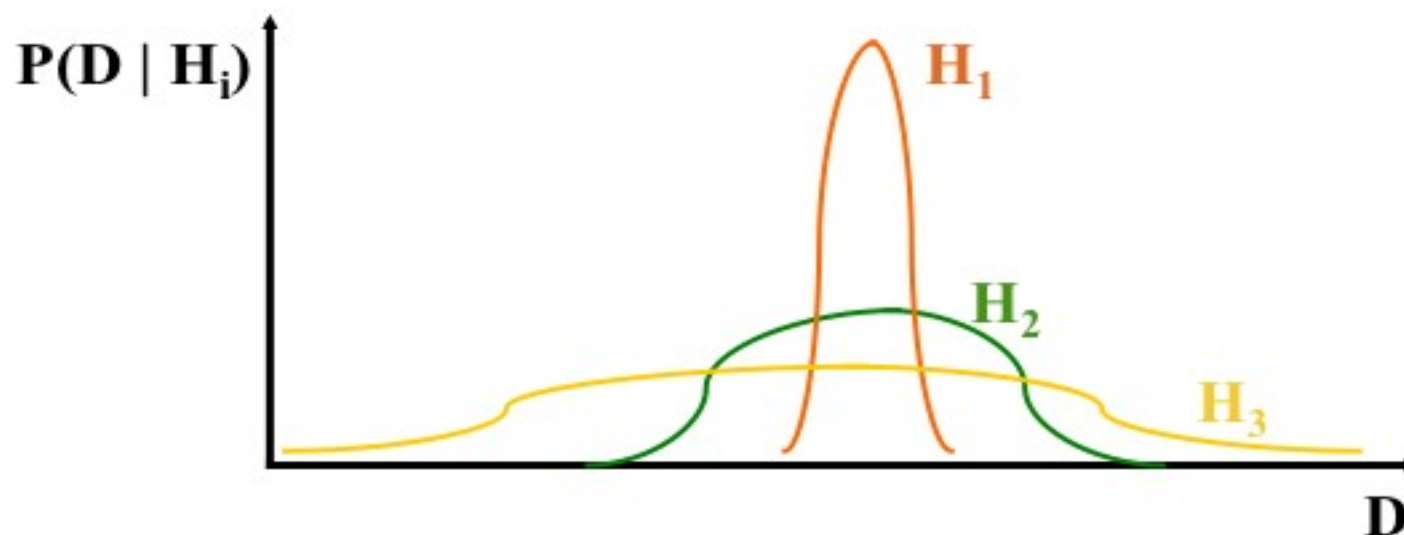


# Lessons learned

- Classical statistics and the Bayesian approach may give contradictory results
  - Using a fixed P-value threshold is problematic as any null hypothesis can be rejected with sufficient amount of data
  - The Bayesian approach compares models and does not aim at an “absolute” estimate of the goodness of the models
- Bayesian model selection depends heavily on the priors selected
  - However, the process is completely transparent and suspicious results can be criticized based on the selected priors
  - Moreover, the impact of the prior can be easily controlled with respect to the amount of available data
- The issue of determining non-informative priors is controversial
  - Reference priors
  - Normalized maximum likelihood & MDL (see [www.mdl-research.org](http://www.mdl-research.org))

## On Bayes factor and Occam's razor

- The marginal likelihood (the “evidence”)  $P(D | H)$  yields a probability distribution (or density) over all the possible data sets  $D$ .
- Complex models can predict well many different data sets, so they need to spread the probability mass over a wide region of models



# Hyperparameters in more complex cases

- Bad news: the BDeu score seems to be quite sensitive to the equivalent sample size (Silander & Myllymäki, UAI'2007)

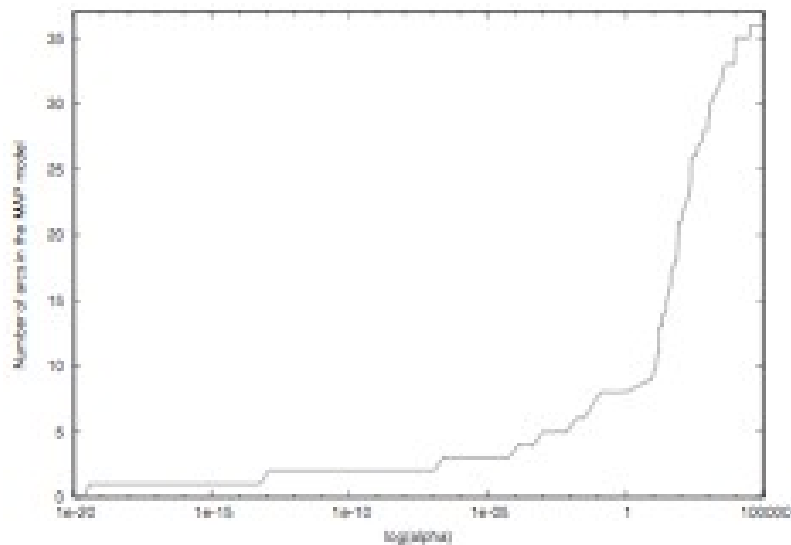


Figure 1: Number of arcs in the BDeu optimal network for the Yeast data as a function of  $\alpha$ .

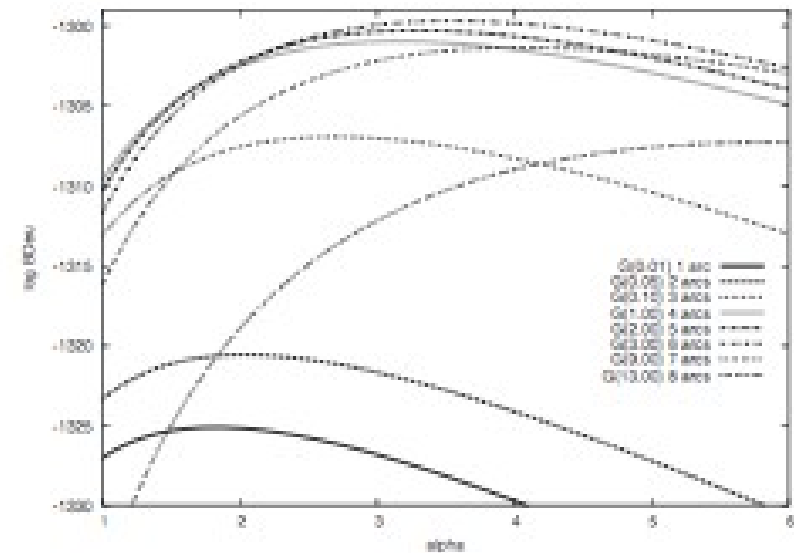


Figure 2: BDeu scores of different MAP models for the Liver data as a function of  $\alpha$ .

# So which prior to use?

- An open issue
- One solution: use the "priorless" Normalized Maximum Likelihood approach
- A more Bayesian solution: use the Jeffreys prior
  - Can be formulated in the Bayesian network framework (Kontkanen et al., 2000), but nobody has produced software for computing it in practice (good topic for your thesis!)
  - B-Course:  $\alpha = \frac{1}{2n} \sum_{i=1}^n r_i$



# Learning the structure: search

# Learning the structure when each node has at most one parent

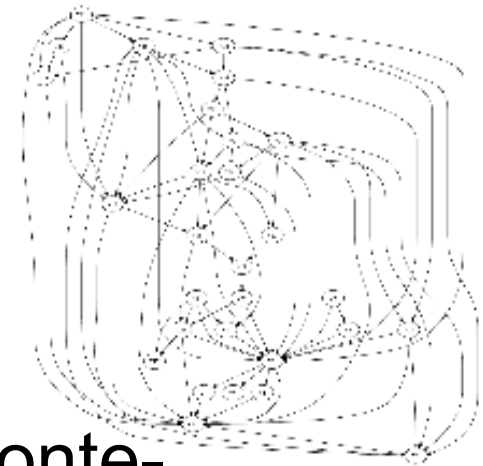
- The BD score is *decomposable*:

$$\begin{aligned} \max_M P(D|M) &= \max_M \prod_i P(X_i[D] | Pa_i^M[D]) \\ &= \min_M \sum_i f_D(X_i, Pa_i^M), \\ \text{where } f_D(X_i, Pa_i^M) &= \log P(X_i[D] | Pa_i^M[D])^{-1} \end{aligned}$$

- For trees (or forests), can use the **minimum spanning tree** algorithm (see Chow & Liu, 1968)

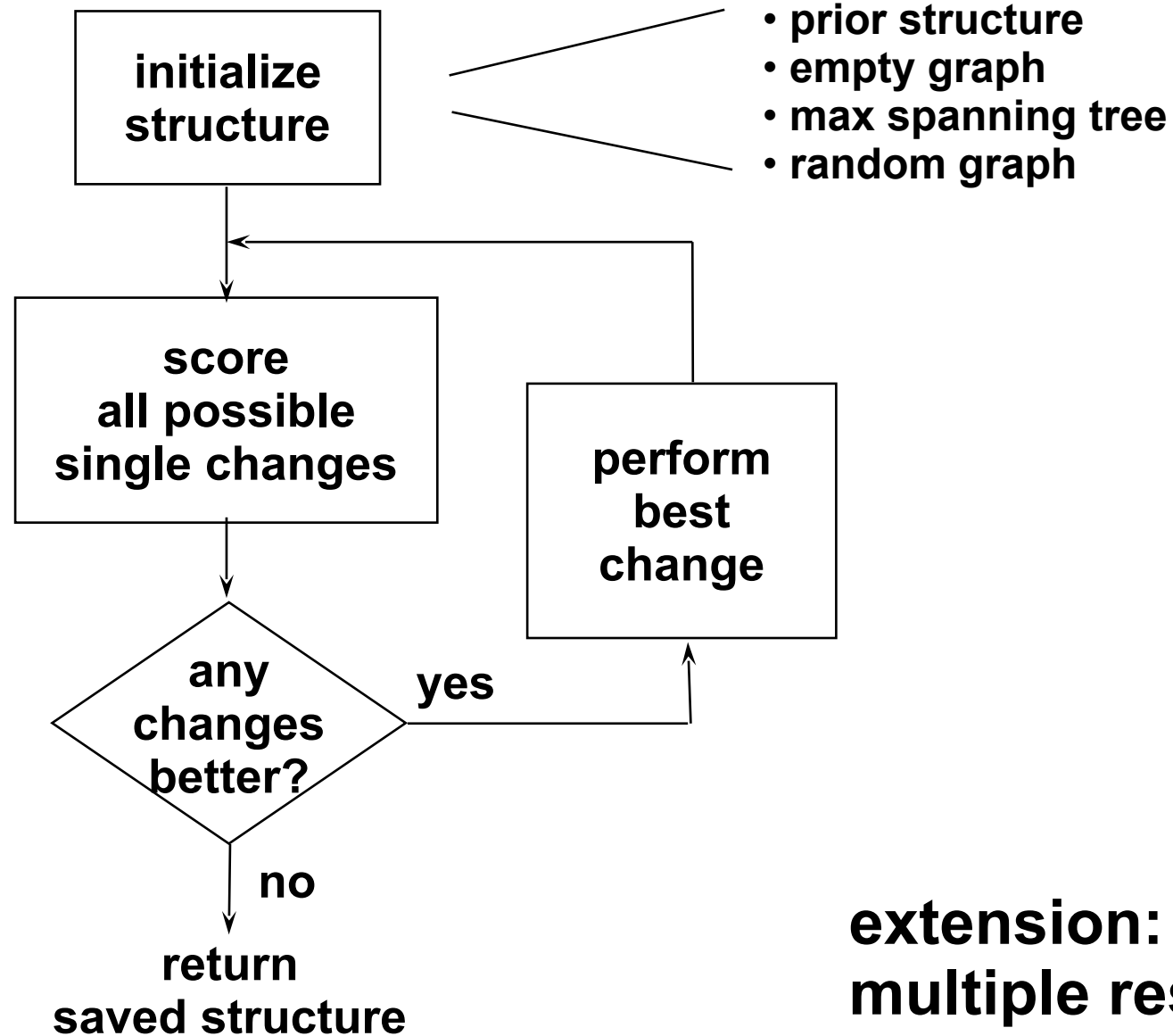
# The General Case

- Finding the best structure is NP-hard, if max. number of parents  $> 1$  (Chickering)
- New dynamic programming solutions work up to  $\sim 30$  variables (Silander & Myllymäki, UAI'2006)
- Heuristics:
  - Greedy bottom-up/top-down
  - Stochastic greedy (with restarts)
  - Simulated annealing and other Monte-Carlo approaches

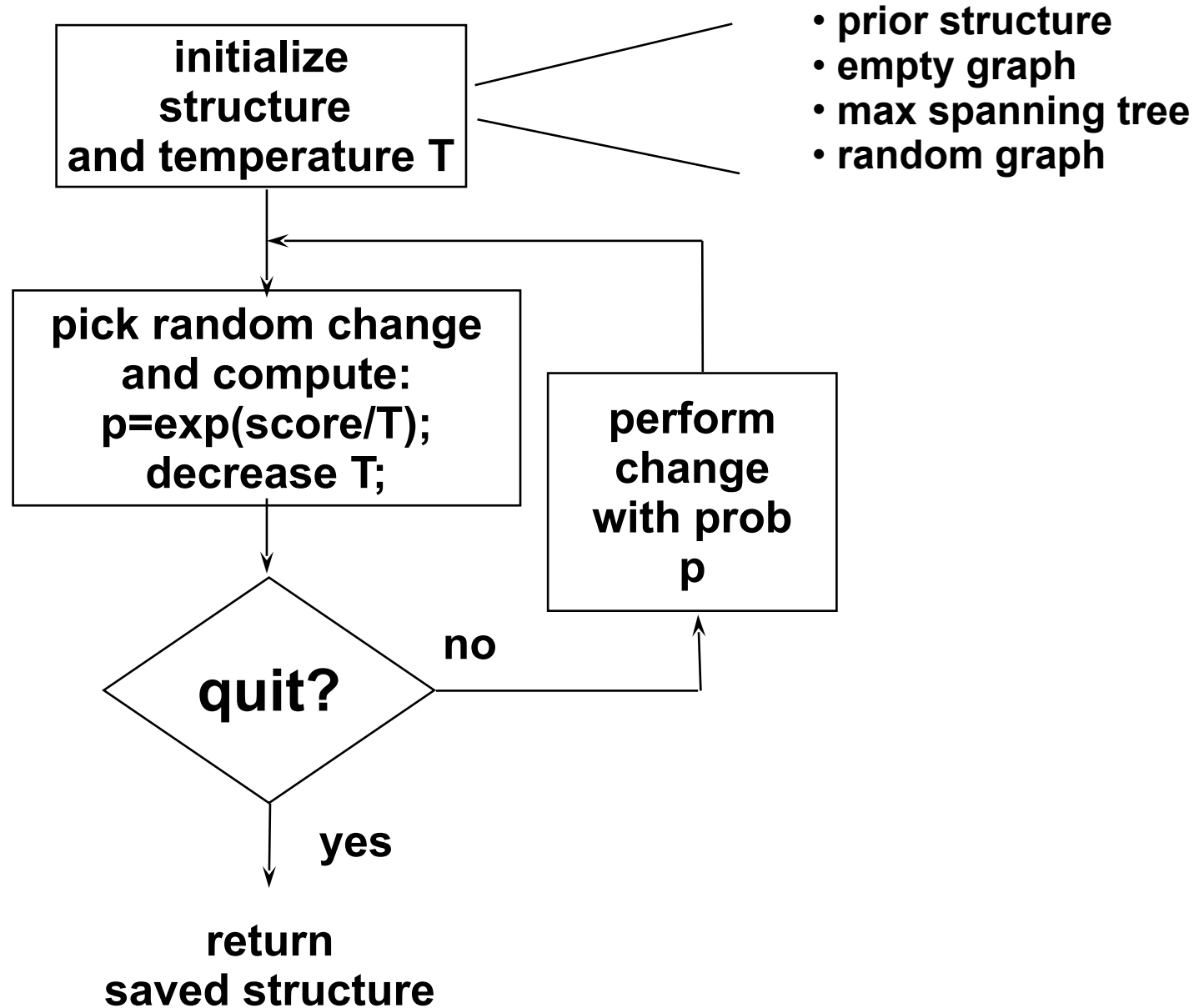




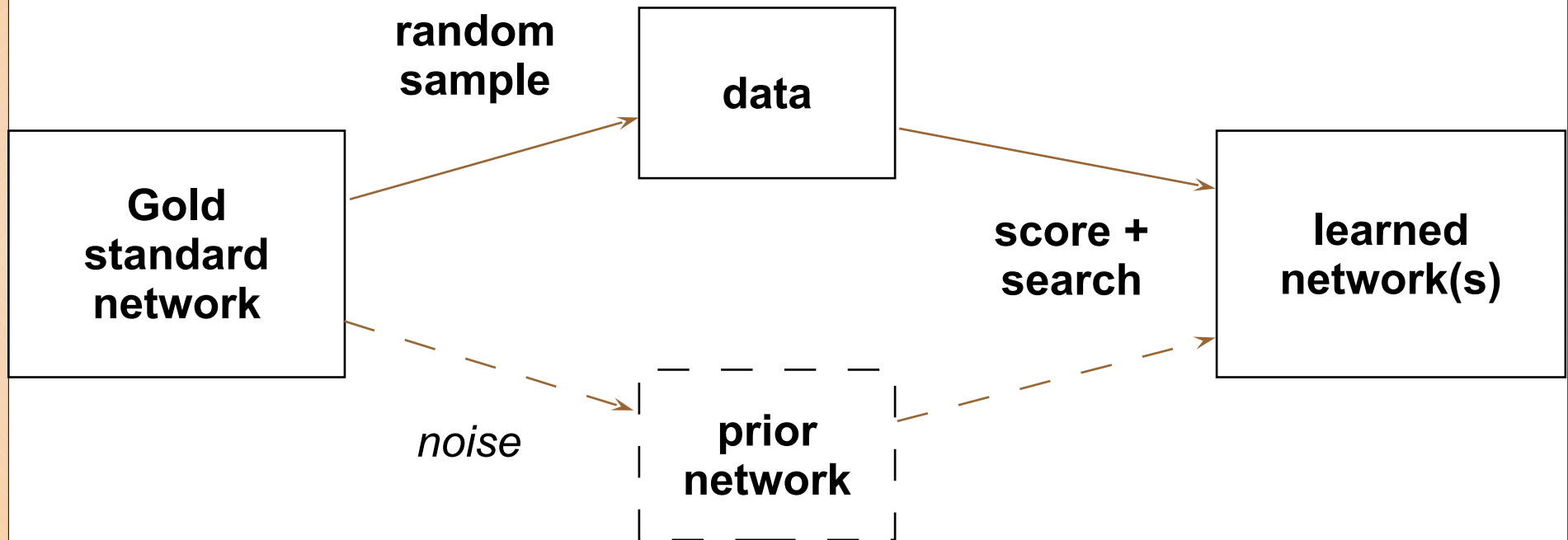
# Local Search



# Simulated Annealing



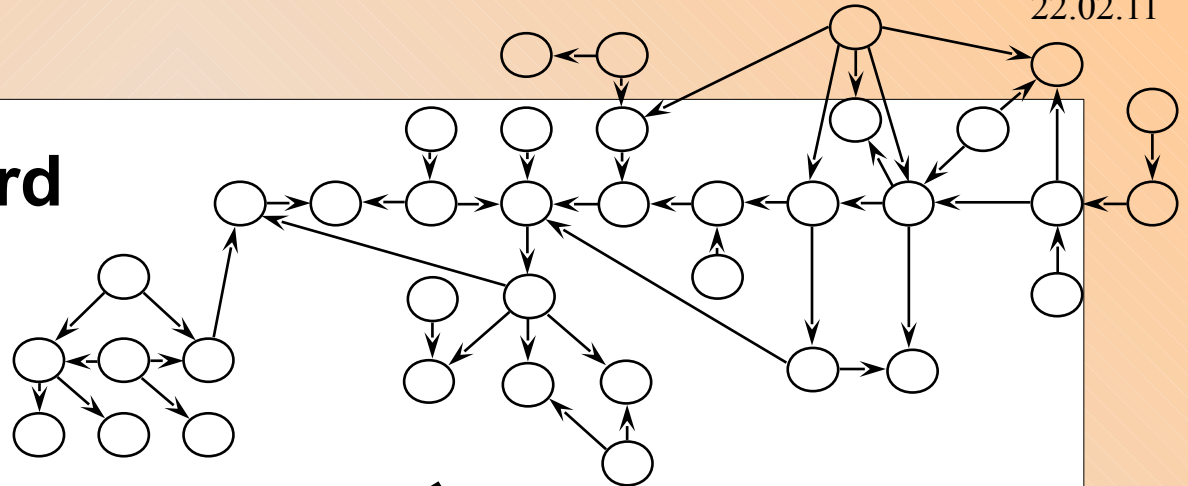
# Evaluation Methodology



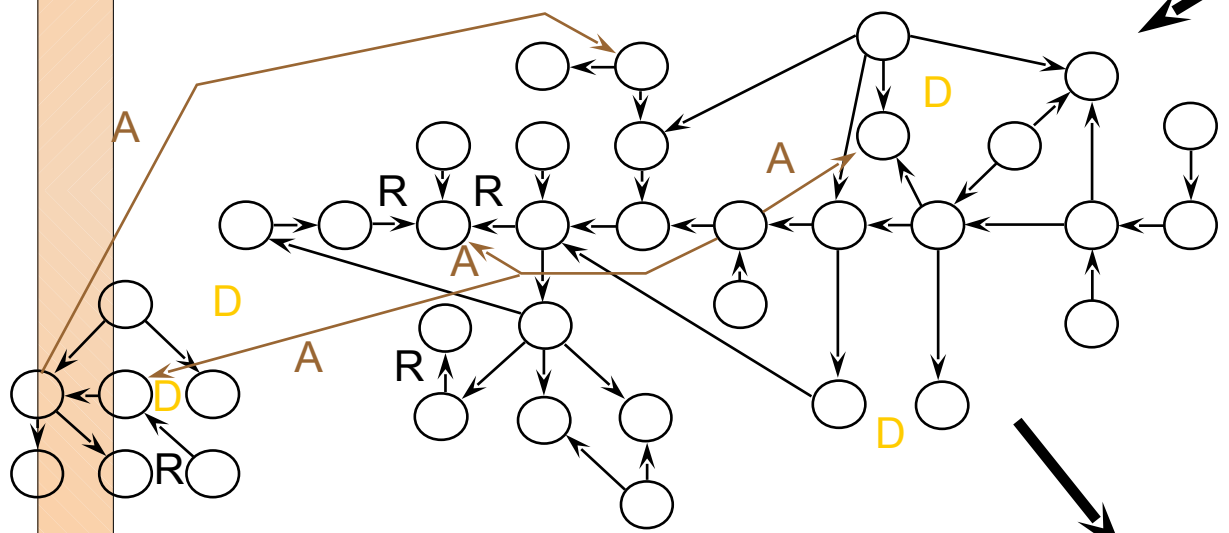
Measures of utility of learned network:

- Cross Entropy (Gold standard network, learned network)
- Structural difference (e.g. #missing arcs, extra arcs, reversed arcs,...)

# Gold Standard



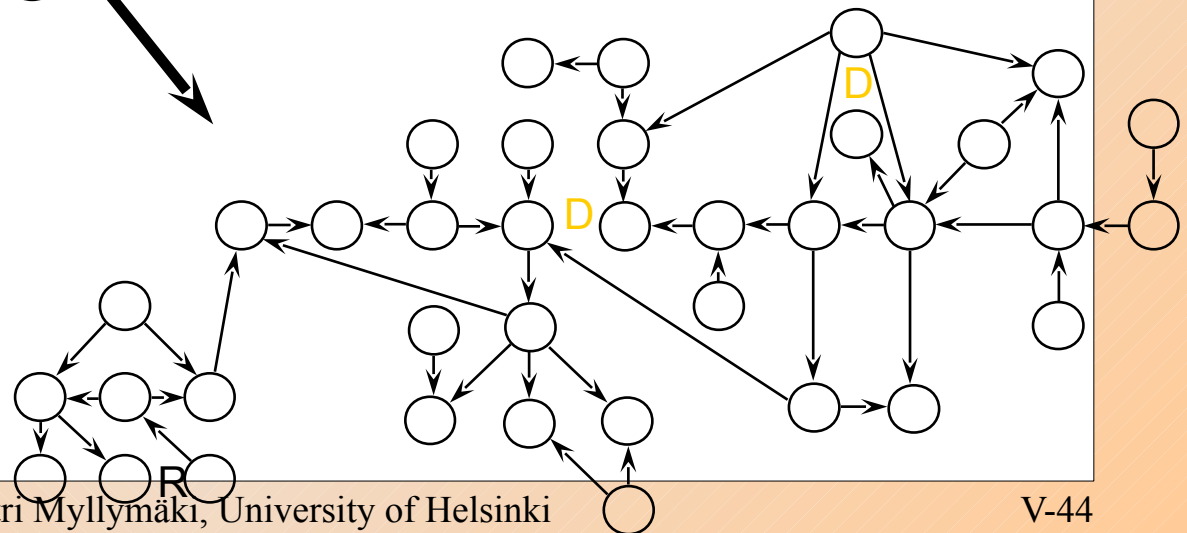
# Prior Network



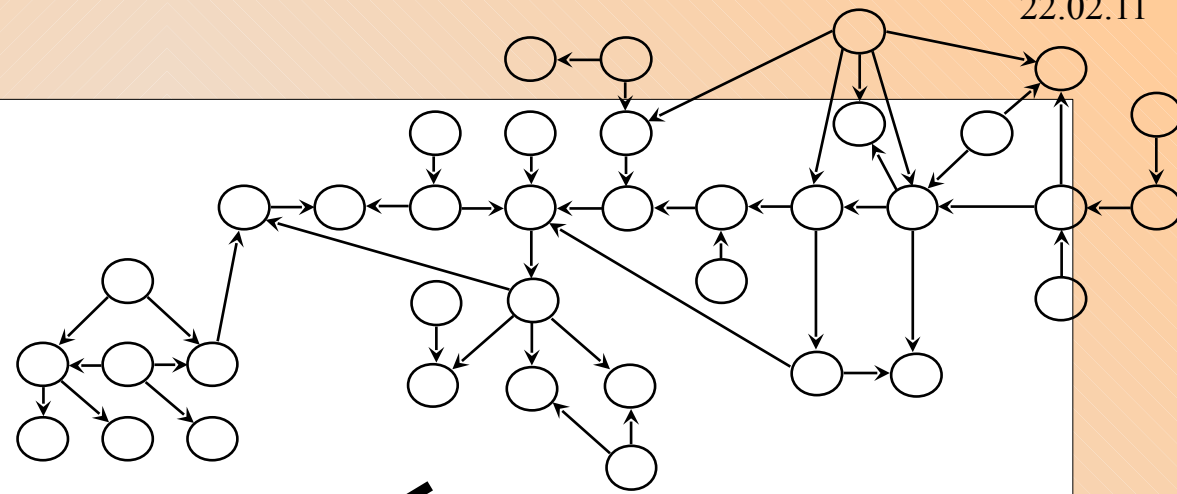
# Data



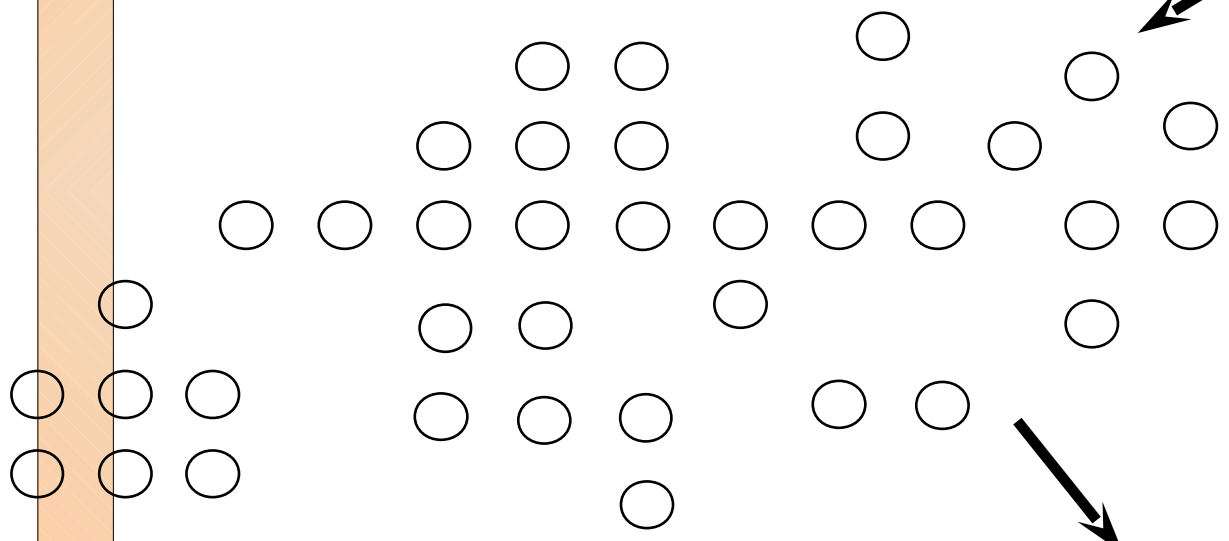
# Learned Network



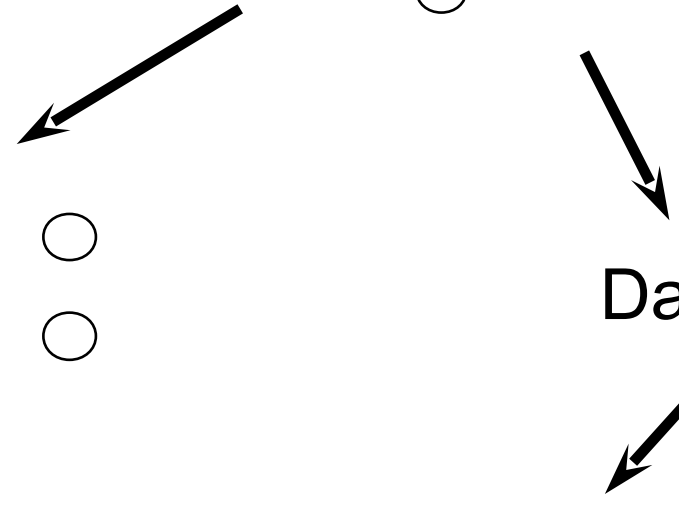
Gold Standard



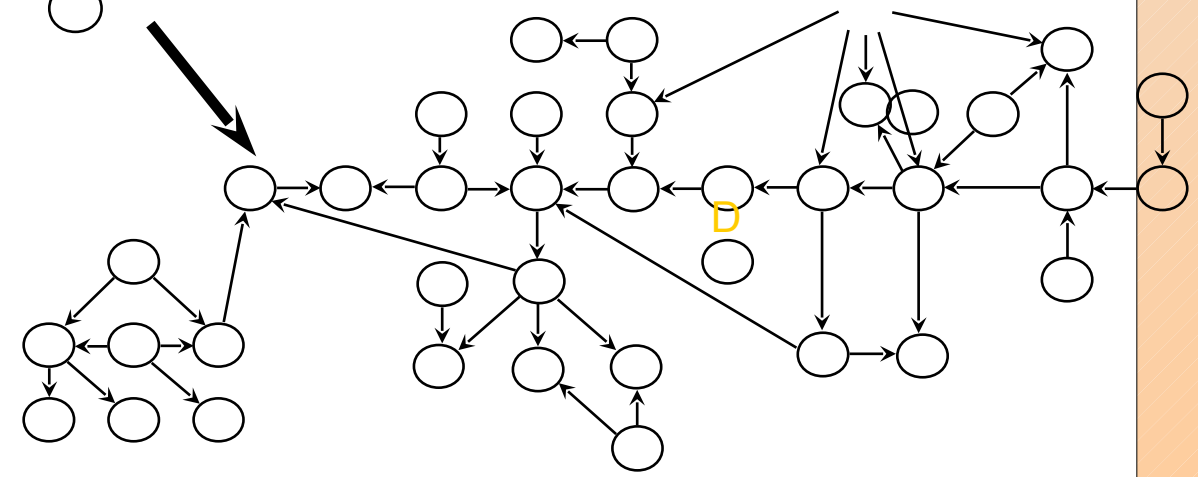
Prior



Data



Learned Network



# Problems with the Gold standard methodology

- Structural criteria may not properly reflect the quality of the result (e.g., the relevance of an extra arc depends on the parameters)
- Cross-entropy (Kullback-Leibler distance) hard to compute
- With small data samples, what is the "correct" answer? Why should the learned network be like the generating network?
- Are there better evaluation strategies? How about predictive performance?