Reasoning and Decision-Making under Uncertainty

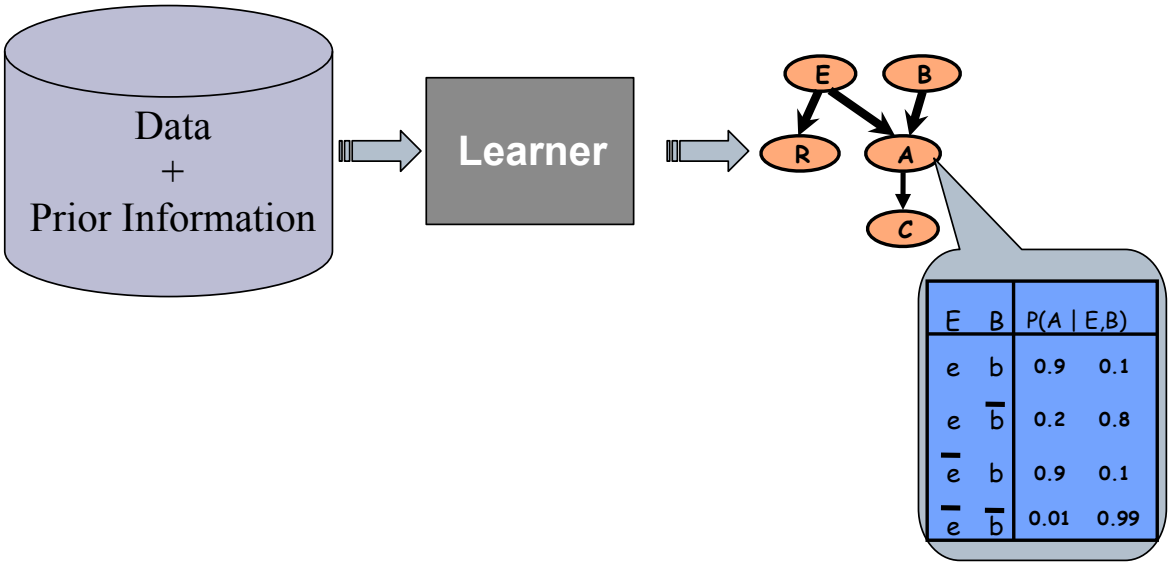
12. Session:

Learning Bayesian Models

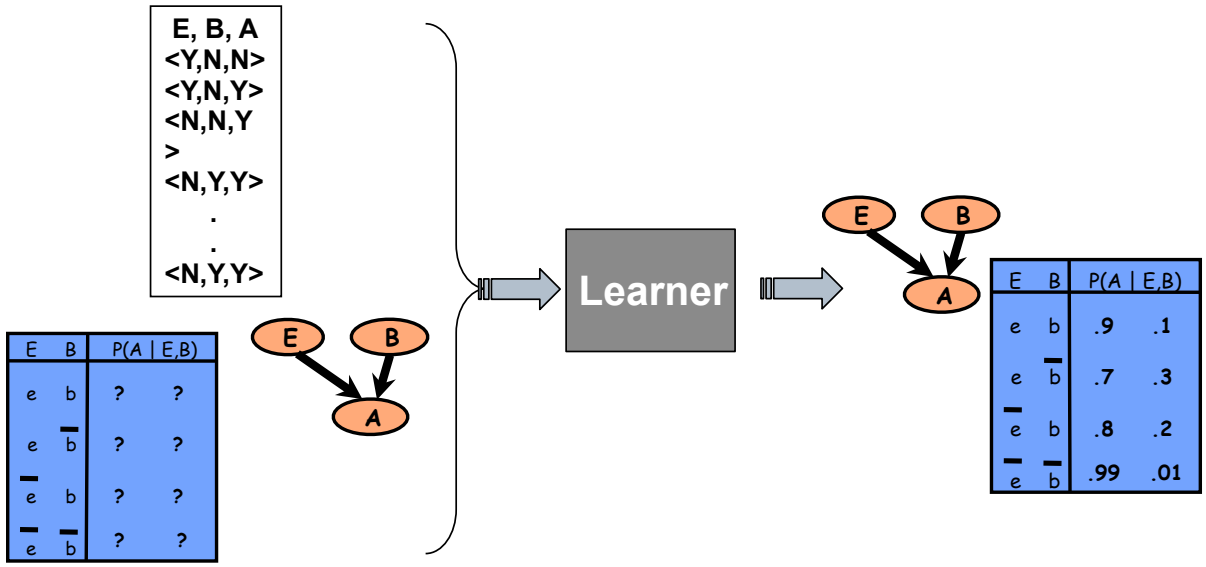
Prof. Dr.-Ing. Stefan Kopp
Center of Excellence „Cognitive Interaction Technology“
AG Sociable Agents

D. Barber (2012), Bayesian Reasoning and Machine Learning, Ch. 9 + 11

Learning Bayesian networks



Known Structure, Complete Data



Network structure is specified, assuming a distribution model, the learner needs to estimate its parameters

Learning as inference over variables

For a distribution $p(x|\theta)$ and data $\mathbf{X}=\{x1,...,xN\}$, learning corresponds to inferring the parameter θ that best explains the data $\mathbf{X}$. Using Bayes:

posterior $p(\theta|\mathbf{X})$ = likelihood $p(\mathbf{X}|\theta)$ * prior $p(\theta)$ / evidence $p(\mathbf{X})$

- Bayesian methods: examine posterior $p(\theta|x) \propto p(\mathbf{X}|\theta)\ p(\theta)$. This gives rise to a distribution over θ.
- Maximum a posteriori: $\theta^{MAP}=\text{argmax}_\theta\ p(\theta|\mathbf{X})$
- Maximum likelihood: Under a flat prior $p(\theta)$=const. the MAP solution is equivalent to setting θ to the value that maximizes the likelihood of observing the data: $\theta^{ML}=\text{argmax}_\theta\ p(\mathbf{X}|\theta)$

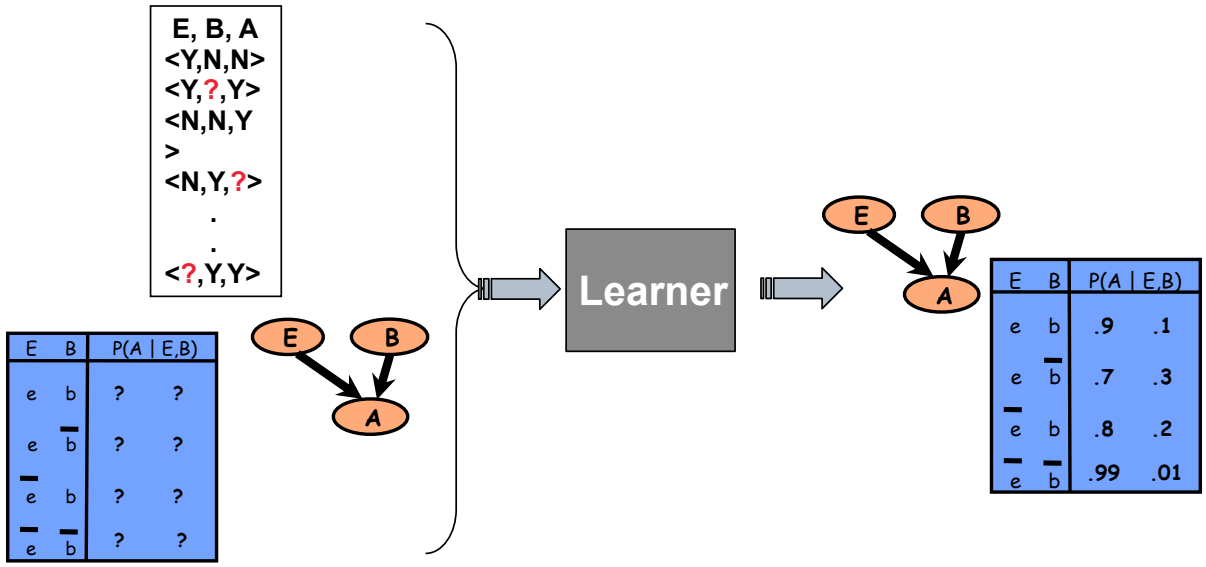
Maximum Likelihood learning

- Assume uniform priors (ok for large data sets)
- Either set tables from sample statistics of the empirical distribution
- ... or choose a parameterized family of distribution models to describe the data and
  - write down likelihood of the data as a function of the parameters
  - write down derivative of the log likelihood w.r.t. each parameter
  - find the parameter values such that the derivatives are zero
  - may be hard/impossible; computational optimization techniques help

If only small data sets available or if we have additional knowledge, we need to place a prior on the tables ➔ Bayesian learning approach

In both cases, only local information need to be considered.

Known Structure, Incomplete Data



Network structure is specified but data contains missing values, learner need to consider assignments to missing values

Incomplete Data

Data is often incomplete, i.e. some variables of interest do not have assigned values. Two possible reasons:

Missing values:

- Some variables are unobserved in some instances
- Missing systematically or missing at random?

Hidden („latent") variables:

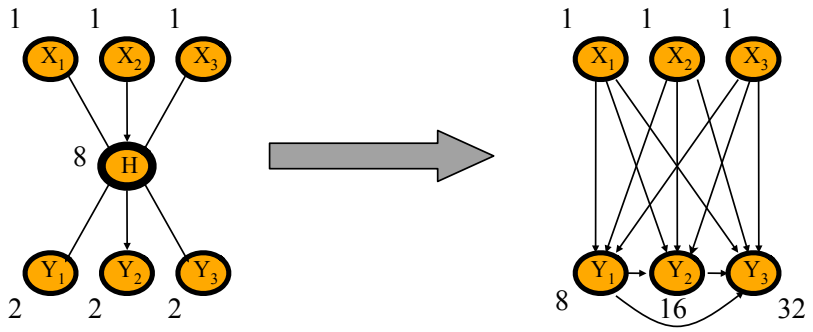
- Some variables are never observed in the data
- We might not even know they exist

Two problems:

- parameter estimation (values of hidden var's ??)
- structure search (existence of hidden var's ??)

Hidden (Latent) Variables

Why should we care about unobserved variables?



17 parameters

59 parameters

Latent var's reduce number of required param's, but complicate the learning problem

Maximum Likelihood

- For hidden variables h, and visible variable v we still have a well defined likelihood

$$p(v|\theta) = \sum_h p(v, h|\theta)$$

- Our task is to find the parameters θ that optimise $p(v|\theta)$.
- This task is more numerically complex than in the case when all the variables are visible.
- Nevertheless, we can perform numerical optimisation using any routine we wish to find θ.
- The Expectation-Maximisation algorithm is an alternative optimisation algorithm that can be very useful in producing simple and elegant updates for θ that converge to a local optimum.
- Just to hammer this home: We don't 'need' the EM algorithm, but it can be very handy.

Expectation Maximization (EM)

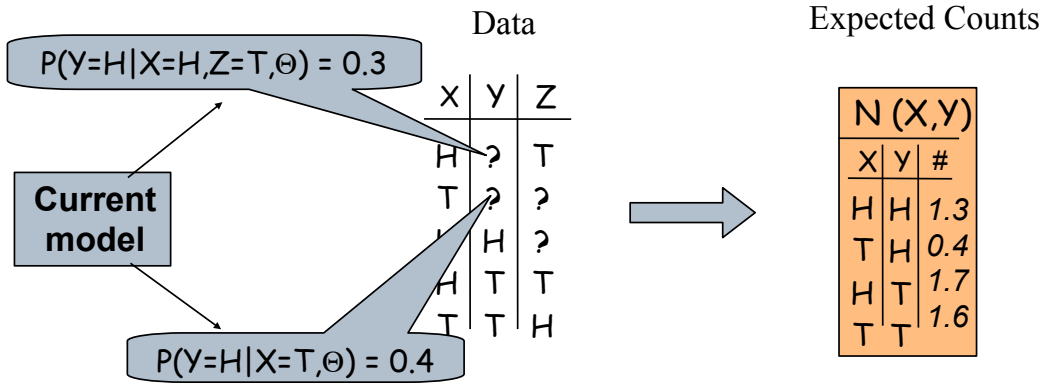
A general purpose method for learning from incomplete data

Idea:

- If we had all variables, we could estimate the parameters
- Let's *pretend* we know all parameters of the model
- We „complete" counts using probabilistic inference from the model, based on our current parameter assignment
- Then use completed variables as if real, to re-estimate (refit) the parameters to the data thereby infering distr. over hidden var's
- Iterate this until parameter(s) or likelihood converge

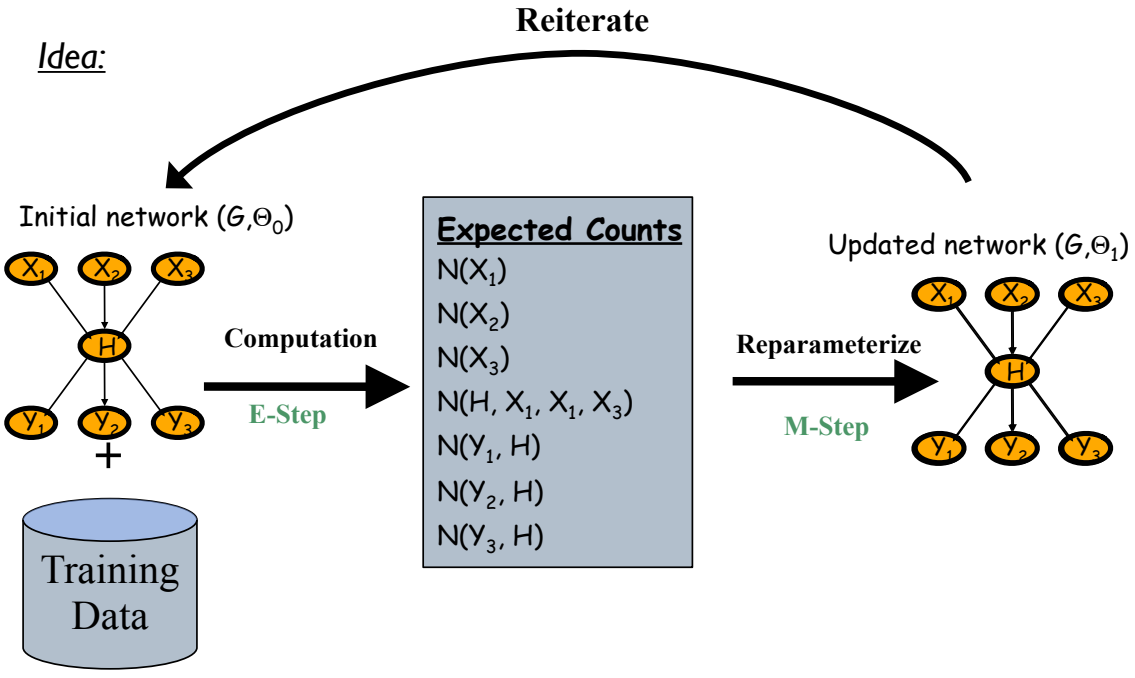
Expectation Maximization (EM)

Idea:



Expectation Maximization (EM)

Idea:



Variational Expectation Maximization (V-EM)

The key feature of the EM algorithm is to form an alternative objective function for which the parameter coupling effect discussed is removed, meaning that individual parameter updates can be achieved, akin to the case of fully observed data. The way this works is to replace the marginal likelihood with a lower bound – it is this lower bound that has the decoupled form.

Consider the Kullback-Leibler divergence between a 'variational' distribution $q(h|v)$ and the parametric model $p(h|v,\theta)$:

$$\mathrm{KL}(q(h|v)|p(h|v,\theta)) \equiv \langle \log q(h|v) - \log p(h|v,\theta) \rangle_{q(h|v)} \geq 0$$

Using Bayes' rule, $p(h|v,\theta) = p(h,v|\theta)/p(v|\theta)$ and the fact that $p(v|\theta)$ does not depend on h,

$$\log p(v|\theta) \geq \underbrace{-\langle \log q(h|v) \rangle_{q(h|v)}}_{\text{Entropy}} + \underbrace{\langle \log p(h,v|\theta) \rangle_{q(h|v)}}_{\text{Energy}}$$

Variational Expectation Maximization (V-EM)

The bound is potentially useful since the energy is similar in form to the fully observed case, except that terms with missing data have their log likelihood weighted by a prefactor.

For i.i.d. data $\mathcal{V} = \{v^1, \ldots, v^N\}$

$$\log p(\mathcal{V}|\theta) \geq -\sum_{n=1}^{N} \langle \log q(h^n|v^n) \rangle_{q(h^n|v^n)} + \sum_{n=1}^{N} \langle \log p(h^n, v^n|\theta) \rangle_{q(h^n|v^n)}$$

E-step For fixed θ, find the distributions $q(h^n|v^n)$ that maximise the bound.

M-step For fixed $\{q(h^n|v^n), n = 1, \ldots, N\}$, find the parameters θ that maximise the bound.

Classical EM

$q(h^n|v^n) = p(h^n|v^n,\theta)$

Expectation Maximization (EM)

Algorithm:

Algorithm 11.1 Expectation maximisation. Compute maximum likelihood value for data with hidden variables. Input: a distribution $p(x|\theta)$ and dataset $\mathcal{V}$. Returns ML candidate θ.

1: $t = 0$ ▷ Iteration counter
2: Choose an initial setting for the parameters θ^0. ▷ Initialisation
3: **while** θ not converged (or likelihood not converged) **do**
4: $t \leftarrow t + 1$
5: **for** $n = 1$ to N **do** ▷ Run over all datapoints
6: $q_t^n(h^n|v^n) = p(h^n|v^n, \theta^{t-1})$ ▷ E-step
7: **end for**
8: $\theta^t = \arg\max_\theta \sum_{n=1}^N \langle \log p(h^n, v^n|\theta) \rangle_{q_t^n(h^n|v^n)}$ ▷ M-step
9: **end while**
10: **return** θ^t ▷ The max likelihood parameter estimate

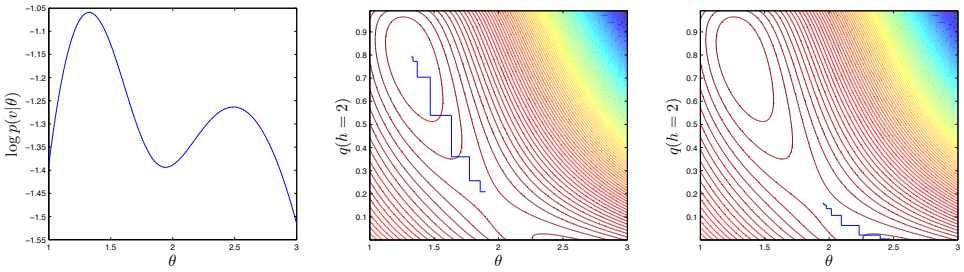
Expectation Maximization (EM)

Example: The model is on a single visible variable v and single two-state hidden variable $h \in \{1, 2\}$. We define a model $p(v, h) = p(v|h)p(h)$ with

$$p(v|h, \theta) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(v-\theta h)^2}$$

and $p(h = 1) = p(h = 2) = 0.5$. For an observation $v = 2.75$ and $\sigma^2 = 0.5$ our interest is to find the parameter θ that optimises the likelihood

$$p(v = 2.75|\theta) = \frac{1}{2\sqrt{\pi}} \sum_{h=1,2} e^{-(2.75-\theta h)^2}$$

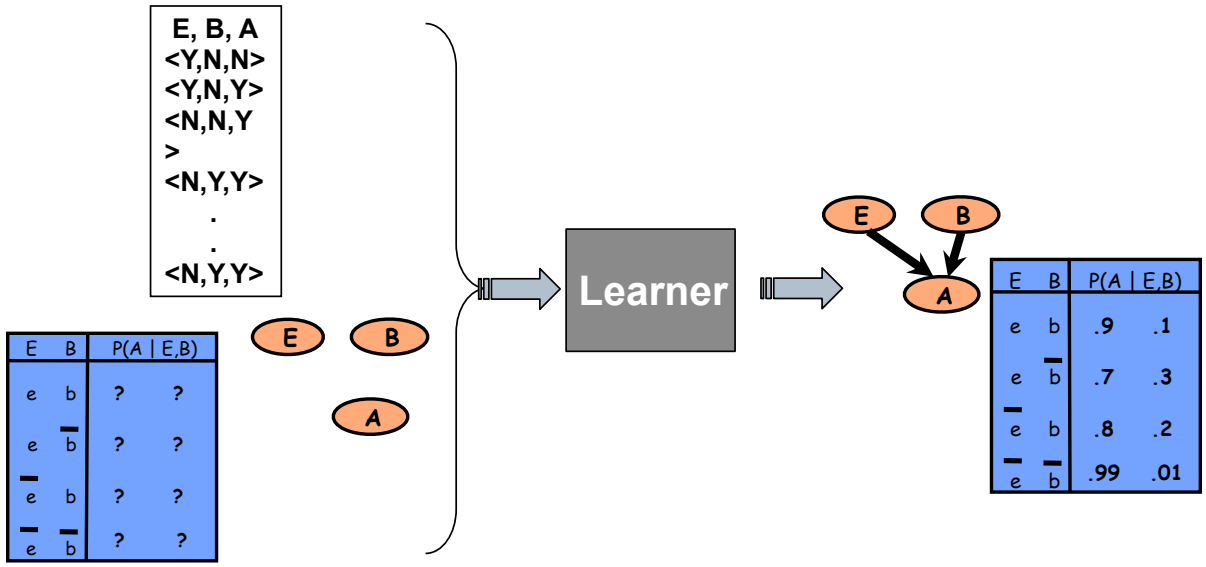


Expectation Maximization (EM)

Remarks:

- EM is easy to implement and can make large jumps in parameter space, particulary in initial iterations
- EM monotonically increases the marginal likelihood (not only the lower bound)
- Convergence of EM can be slow, particularly when #missing observations > #visible observations
  - often combined with gradient-based methods to improve convergence
- EM can run into local optimum, solution often depends on initialization
- Straightforward application of EM training to learn the CPTs in a Bayesian network under missing data (see *Barber Chapt. 11*)

Unknown Structure, Complete Data



Network structure is *not* specified, learner needs to select graph structure & estimate parameters

Structure Learning

Lack of a priori independence knowledge

We assume we have a dataset, but don't know the independence assumptions we should make.

No missing data

For simplicity, we assume that the dataset is complete (there are no missing observations).

(almost) Complete ignorance

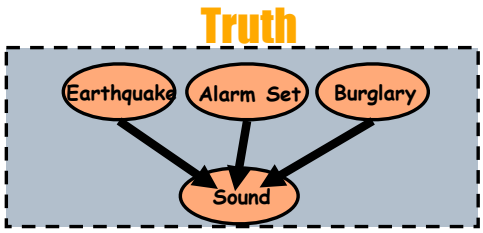
One could also consider the case of knowing some conditional independence assumptions, but not all. For simplicity, we assume that none are known.

Difficulty

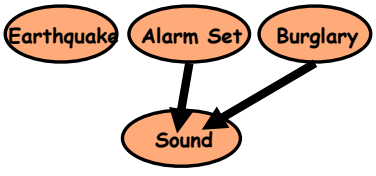
Number of DAGs on N nodes is at least

$$\prod_{n=1}^{N} 2^n = 2^{N(N-1)/2}$$

Why Struggle for Accurate Structure?



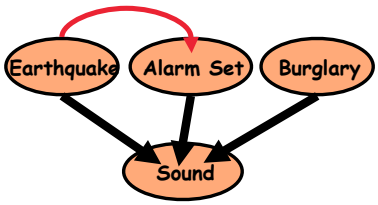
Missing an arc



Cannot be compensated for by fitting parameters

Wrong assumptions about domain structure

Adding an arc



Increased parameters to be estimated

Wrong assumptions about domain structure

How to learn a causal structure?

A Bayesian Network represents a causal model, i.e. finding a graph structure means to detect causal structure

- but, statistical analysis is driven by correlation, not causation!

How to detect cause-effect relationships?

- cues used by humans: temporal and statistical (cond. indep.)
- careful manipulations of variables to test effects (experiments)

Normally not possible from mere observation in an environment!

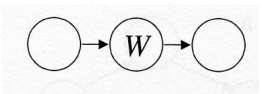
- can look for patterns that are characteristic of causal relations
- but problems (cf. Pearl (2009), Chapt. 2):
  - latent variables, parents (Markov condition), spurious associations, non-temporal data, non-uniqueness of structure, minimal models, stability of distributions, etc.

How to learn a causal structure?

Recall: There are three types of causal structures in a graph, associated with probabilistic conditional independence relations

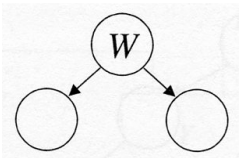
- „valves" that can be either open or closed to block a path, implying conditional independence
- „sufficient" causality to explain the observations (causes for effects)

Sequential, chain



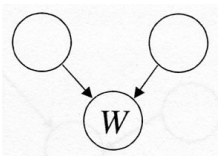
Blocked if *W* is given

Divergent, fork



Blocked if *W* is given

Convergent, *collider*, inverse fork



Blocked if W is not given

Need heuristic search

Global search

Traverse space of possible structures looking for high-scoring structures

- operators make small changes to structure (e.g. add/remove edges)
- Greedy hill-climbing, Best first search, Simulated Annealing

Local Search

Start with a network (empty vs. best tree vs. random)

- operators evaluate possible local changes and apply change based on maximum score
- stop if no modification improves score, but beware of pitfalls!

In practice, search is relatively fast (~100 vars in ~2-5 min)
Adding randomness to search is critical

PC algorithm: Learning the skeleton

Removing links

- Start with a complete skeleton G
- Test all pairs $x \perp\!\!\!\perp y$? If an x and y pair are deemed independent then the link $x - y$ is removed from G.
- In the next round, for the remaining graph, one examines each $x - y$ link and conditions on a single neighbour z of x. If $x \perp\!\!\!\perp y | z$ then remove the link $x - y$.
- At each subsequent round the number of neighbours in the conditioning set is increased by one and all $x \perp\!\!\!\perp y | \mathcal{Z}$ are tested.

Storing conditions of independence

Whenever a (conditional) independence is found $x \perp\!\!\!\perp y | \mathcal{Z}$, then these conditioning variables are stored in a set $\mathcal{S}_{x,y} = \mathcal{Z}$ (this could be the empty set).

Assessing independence hypotheses

Given a dataset of observations, how can we decide if two variables x and y are independent?

Mutual Information

We can form the empirical distributions $p(x)$ and $p(y)$ and $p(x, y)$. Define the mutual information

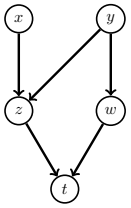
$$MI \equiv \mathrm{KL}(p(x, y)|p(x)p(y)) \geq 0$$

If $MI = 0$ then x and y are independent.

The classical approach is to use a hypothesis test, assuming that MI is chi-square distributed. This doesn't work well for small numbers of observations.

A Bayesian approach to testing for independence can be made by comparing the likelihood of the data under the independence hypothesis, versus the likelihood under the dependent hypothesis.

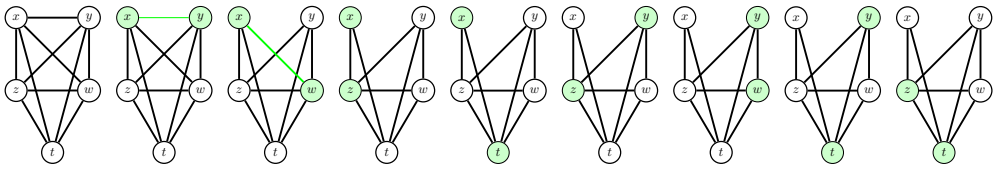
PC algorithm



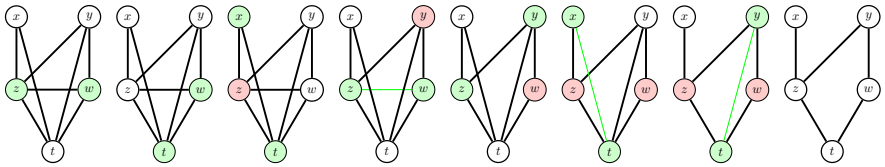
1. round (c-l): $x \perp\!\!\!\perp y | \emptyset$
2. round (m-o): $x \perp\!\!\!\perp y | z$
3. round (p,q): $x \perp\!\!\!\perp y | \{a, b\}$

terminate since no nodes with more than 3 neighbors left

$S_{x,y} = \emptyset, S_{x,w} = \emptyset, S_{z,w} = y, S_{x,t} = \{z, w\}, S_{y,t} = \{z, w\}$

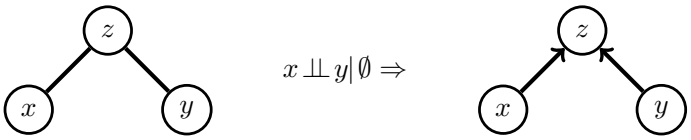


(b) (c) (d) (e) (f) (g) (h) (i) (j)

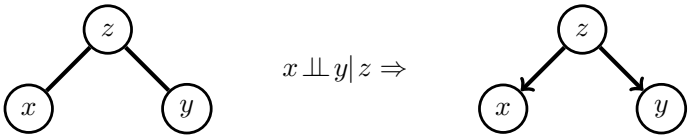


(k) (l) (m) (n) (o) (p) (q) (r)

Skeleton orienting



If x is (unconditionally) independent of y, it must be that z is a collider since otherwise marginalising over z would introduce a dependence between x and y.



If x is independent of y conditioned on z, z must not be a collider. Any other orientation is appropriate.

Skeleton orienting

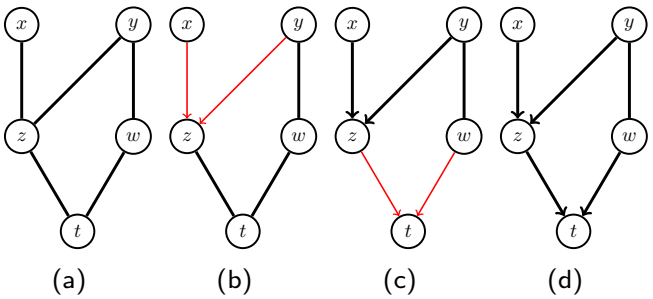
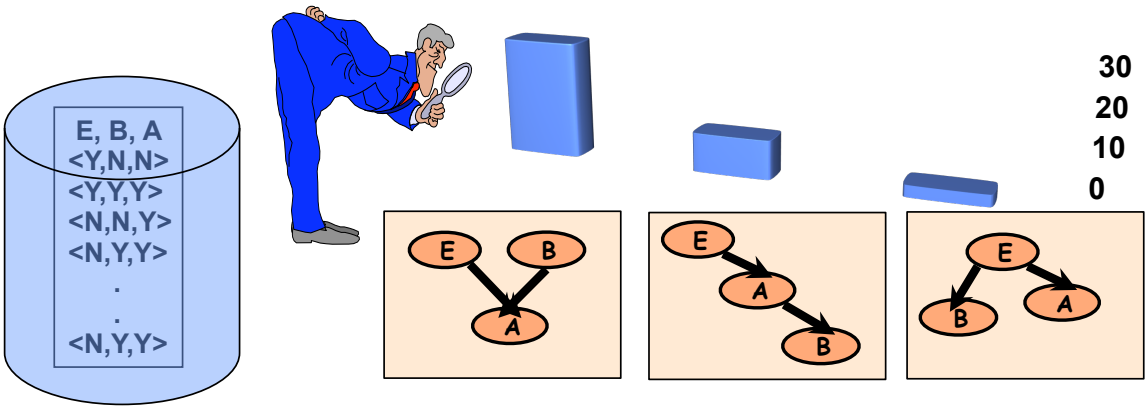


Figure: Skeleton orientation algorithm. **(a)**: The skeleton along with $S_{x,y} = \emptyset, S_{x,w} = \emptyset, S_{z,w} = y, S_{x,t} = \{z, w\}, S_{y,t} = \{z, w\}$. **(b)**: $z \notin S_{x,y}$, so form collider. **(c)**: $t \notin S_{z,w}$, so form collider. **(d)**: Final partially oriented DAG. The remaining edge may be oriented as desired, without violating the DAG condition.

Network scoring

Idea: define scoring function that evaluates how well a structure matches the data and optimize structure to maximize this score



Network scoring

Local versus global methods

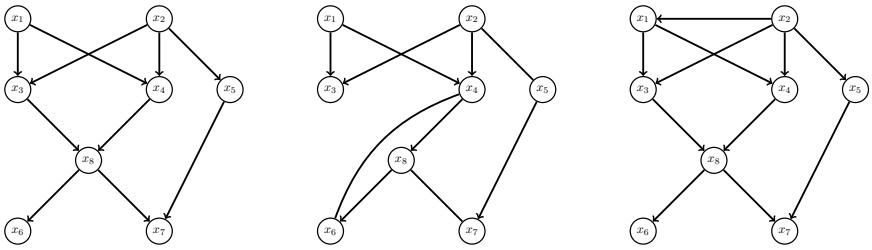
The PC algorithm is local in the sense that links are added based on the evidence for a link on the basis of local data. In a global method, a link is added based on how well that resulting distribution fits the data.

A probabilistic approach

- In a probabilistic context, given a model structure M, we wish to compute $p(M|\mathcal{D}) \propto p(\mathcal{D}|M)p(M)$.
- We have to first 'fit' each model with parameters θ, $p(\mathcal{V}|\theta, M)$ to the data $\mathcal{D}$. If we do this using Maximum Likelihood alone, with no constraints on θ, we will favour that model M with the most complex structure.
- This can be remedied by using the Bayesian technique

$$p(\mathcal{D}|M) = \int_{\theta} p(\mathcal{D}|\theta, M)p(\theta|M)$$

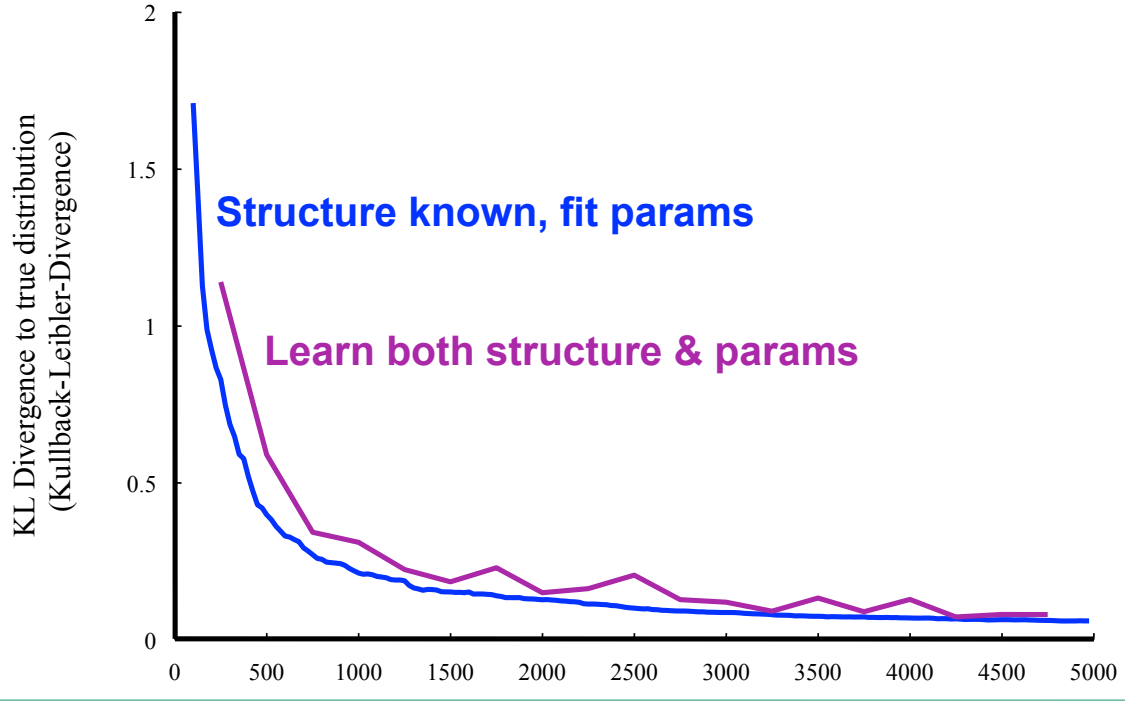
Comparison



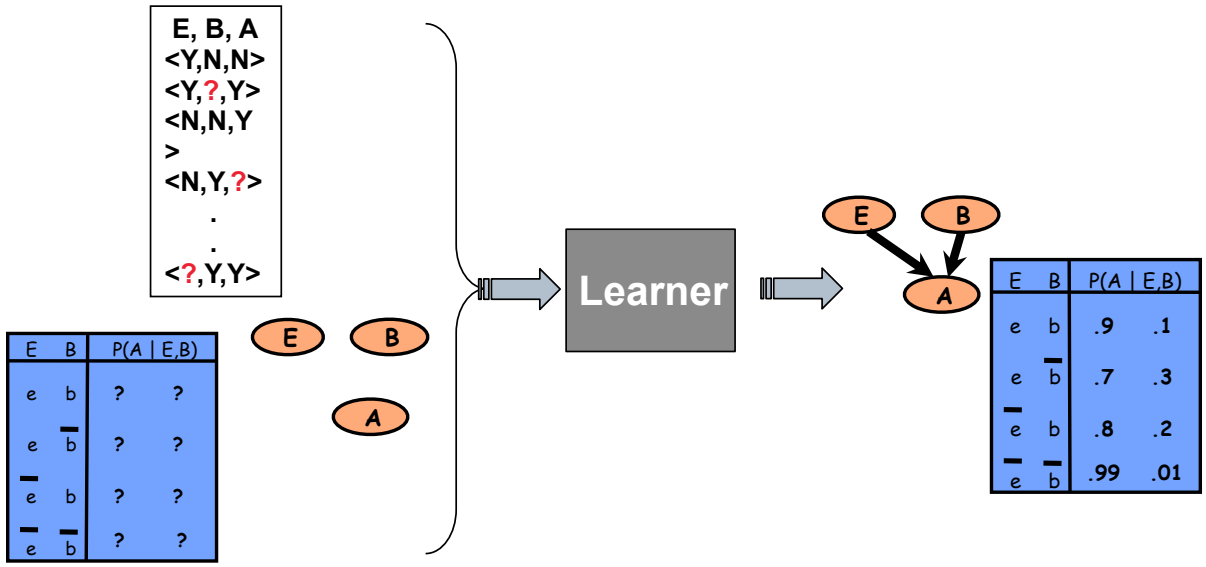
(a) (b) (c)

Figure: Learning the structure of a Bayesian network. **(a)**: The correct structure in which all variables are binary. The ancestral order is $x_2, x_1, x_5, x_4, x_3, x_8, x_7, x_6$. The dataset is formed from 1000 samples from this network. **(b)**: The learned structure based on the PC algorithm using the Bayesian empirical conditional independence test. Undirected edges may be oriented arbitrarily (provided the graph remains acyclic). **(c)**: The learned structure based on the Bayes Dirichlet network scoring method (assuming we know the ancestral order and have maximally two parents).

Learning in Practice: Alarm domain



Unknown Structure, Incomplete Data



Network structure is not specified and data contains missing values, need to select graph structure, estimate parameters, assign missing values

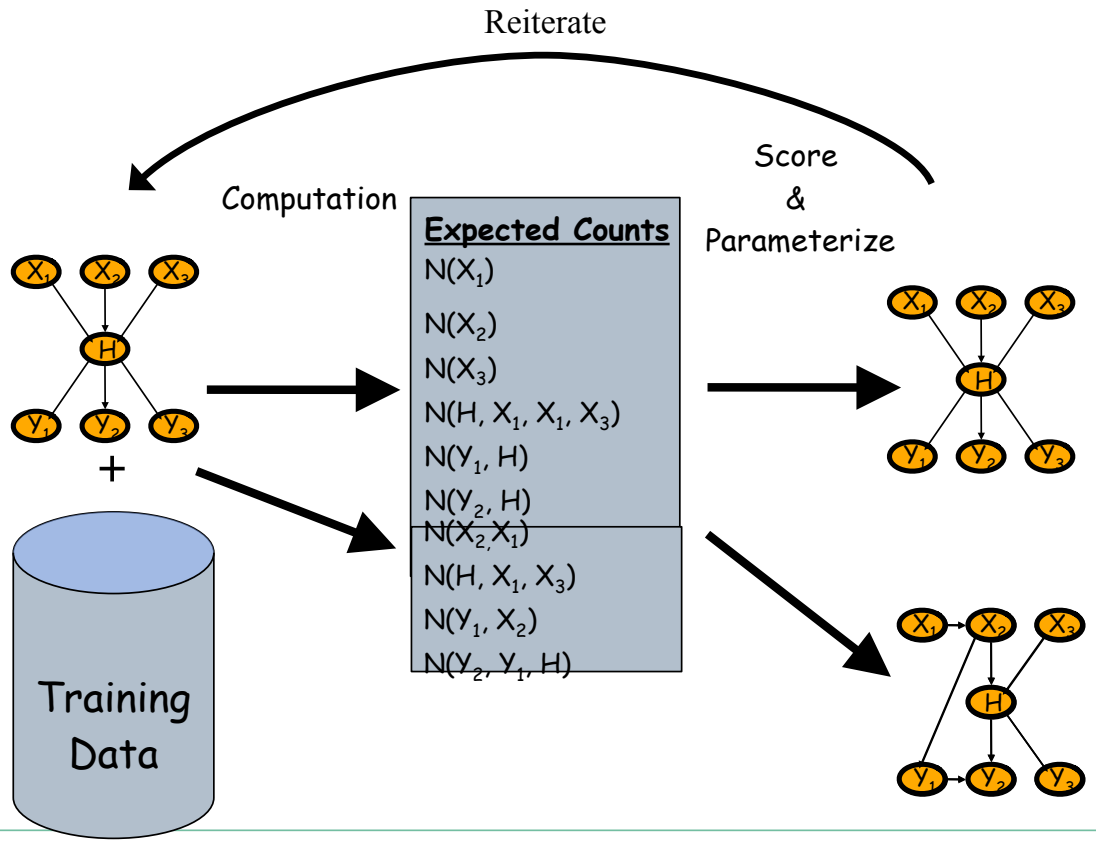
Structural EM

Idea:

- Instead of optimizing the real score, find decomposable alternative score
- Use current model to help evaluate new structures

Outline:

- Perform combined search in (Structure, Parameters)-space
- At each iteration, use current model for finding *either*:
  - better scoring parameters: "parametric" EM step
  - better scoring structure: "structural" EM step



Summary: Statistical Learning

Learn network structure & parameters from data

Parameter Estimation

- Maximum-likelihood estimation (MLE)
- Bayesian estimation when priors available

Model Selection (Structure learning)

- Local tests of independence
- Global structure search as score-based optimization

Learning from incomplete data (missing observations, hidden var's)

- Expectation Maximization (EM)
- Combined structure & parameter search

Want to work with us?

Applying advanced A.I. methods and cognitive models to increase **interaction abilities** of technical systems

Manifold sources of uncertainty:

- complex communication system (language, nonverbal behavior)
- noisy recognition and processing problems
- vague & underspecified meanings, implied meaning
- implicit communication, underlying intentions
- dynamic nature of interpersonal coordination