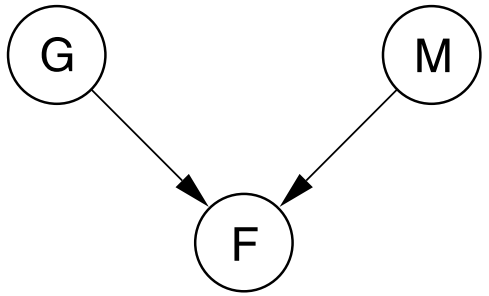
Learning Graphical Models

Prerequisites: Structure vs. Parameters

$A_1 = \mathsf{G}$	$Q_{11} = \phi$
$a_{11} = \mathsf{g}$	
$a_{12} = \overline{\mathsf{g}}$	

$A_2 = \mathsf{M}$	$Q_{21} = \phi$
$a_{12} = \mathsf{m}$	
$a_{22} = \overline{\mathsf{m}}$	



$A_3 = \mathsf{F}$	$Q_{31} = (\mathsf{g}, \mathsf{m})$	$Q_{32} = (\mathsf{g}, \overline{\mathsf{m}})$	$Q_{33} = (\overline{\mathsf{g}}, \mathsf{m})$	$Q_{34} = (\overline{\mathsf{g}}, \overline{\mathsf{m}})$
$a_{31} = \mathsf{f}$				
$a_{32} = \overline{\mathsf{f}}$				

- $V = \{\mathsf{G}, \mathsf{M}, \mathsf{F}\}$
- $\mathrm{dom}(\mathsf{G}) = \{\mathsf{g}, \overline{\mathsf{g}}\}$
- $\mathrm{dom}(\mathsf{M}) = \{\mathsf{m}, \overline{\mathsf{m}}\}$
- $\mathrm{dom}(\mathsf{F}) = \{\mathsf{f}, \overline{\mathsf{f}}\}$

- The potential tables' layout is determined by the graph structure.
- The parameters (i. e. the table entries) can be easily estimated from the database, e. g.:

$$\hat{P}(\mathsf{f} \mid \mathsf{g}, \mathsf{m}) = \frac{\#(\mathsf{F} = \mathsf{f}, \mathsf{G} = \mathsf{g}, \mathsf{M} = \mathsf{m})}{\#(\mathsf{G} = \mathsf{g}, \mathsf{M} = \mathsf{m})}$$

Prerequisites: Likelihood of a Database

Flu G	$\overline{\mathsf{g}}$	$\overline{\mathsf{g}}$	$\overline{\mathsf{g}}$	$\overline{\mathsf{g}}$	g	g	g	g
Malaria M	$\overline{\mathsf{m}}$	$\overline{\mathsf{m}}$	m	m	$\overline{\mathsf{m}}$	$\overline{\mathsf{m}}$	m	m
Fever F	$\overline{\mathsf{f}}$	f	$\overline{\mathsf{f}}$	f	$\overline{\mathsf{f}}$	f	$\overline{\mathsf{f}}$	f
#	34	6	2	8	16	24	0	10

Database D with 100 entries for 3 attributes.

$$P(D \mid G) \quad = \quad \prod_{h=1}^{100} P(c_h \mid G)$$

$$= \overbrace{P(\mathsf{g},\mathsf{m},\mathsf{f}) \cdot \ldots \cdot P(\mathsf{g},\mathsf{m},\mathsf{f})}^{\text{Case 1}\ \ldots\ \text{Case 10}} \quad \cdots \quad \overbrace{P(\overline{\mathsf{g}},\mathsf{m},\mathsf{f}) \cdot \ldots \cdot P(\overline{\mathsf{g}},\mathsf{m},\mathsf{f})}^{\text{Case 51}\ \ldots\ \text{Case 58}} \quad \cdots \quad \overbrace{P(\overline{\mathsf{g}},\overline{\mathsf{m}},\overline{\mathsf{f}}) \cdot \ldots \cdot P(\overline{\mathsf{g}},\overline{\mathsf{m}},\overline{\mathsf{f}})}^{\text{Case 67}\ \ldots\ \text{Case 100}}$$

$$= \overbrace{P(\mathsf{g},\mathsf{m},\mathsf{f})^{10}}^{10 \text{ times}} \quad \cdots \quad \overbrace{P(\overline{\mathsf{g}},\mathsf{m},\mathsf{f})^{8}}^{8 \text{ times}} \quad \cdots \quad \overbrace{P(\overline{\mathsf{g}},\overline{\mathsf{m}},\overline{\mathsf{f}})^{34}}^{34 \text{ times}}$$

$$= P(\mathsf{f} \mid \mathsf{g},\mathsf{m})^{10} P(\mathsf{g})^{10} P(\mathsf{m})^{10} \quad \cdots \quad P(\mathsf{f} \mid \overline{\mathsf{g}},\mathsf{m})^{8} P(\overline{\mathsf{g}})^{8} P(\mathsf{m})^{8} \quad \cdots \quad P(\overline{\mathsf{f}} \mid \overline{\mathsf{g}},\overline{\mathsf{m}})^{34} P(\overline{\mathsf{g}})^{34} P(\overline{\mathsf{m}})^{34}$$

Prerequisites: Likelihood of a Database (2)

$$P(D \mid G) \quad = \quad \prod_{h=1}^{100} P(c_h \mid G)$$

$$= P(\mathsf{f} \mid \mathsf{g}, \mathsf{m})^{10} P(\overline{\mathsf{f}} \mid \mathsf{g}, \mathsf{m})^{0} P(\mathsf{f} \mid \mathsf{g}, \overline{\mathsf{m}})^{24} P(\overline{\mathsf{f}} \mid \mathsf{g}, \overline{\mathsf{m}})^{16}$$

$$\cdot P(\mathsf{f} \mid \overline{\mathsf{g}}, \mathsf{m})^{8} P(\overline{\mathsf{f}} \mid \overline{\mathsf{g}}, \mathsf{m})^{2} P(\mathsf{f} \mid \overline{\mathsf{g}}, \overline{\mathsf{m}})^{6} P(\overline{\mathsf{f}} \mid \overline{\mathsf{g}}, \overline{\mathsf{m}})^{34}$$

$$\cdot P(\mathsf{g})^{50} P(\overline{\mathsf{g}})^{50} P(\mathsf{m})^{20} P(\overline{\mathsf{m}})^{80}$$

The last equation shows the principle of reordering the factors:

- First, we sort by attributes (here: F, G then M).
- Within the same attributes, factors are grouped by the parent attributes' values combinations (here: for F: (g, m), $(\mathsf{g}, \overline{\mathsf{m}})$, $(\overline{\mathsf{g}}, \mathsf{m})$ and $(\overline{\mathsf{g}}, \overline{\mathsf{m}})$).
- Finally, it is sorted by attribute values (here: for F: first f, then $\overline{\mathsf{f}}$).

Prerequisites: Likelihood of a Database (3)

General likelihood of a database D given a DAG G:

$$P(D \mid G) \quad = \quad \prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}}$$

General potential table:

A_i	Q_{i1}	$\cdots$	Q_{ij}	$\cdots$	Q_{iq_i}
a_{i1}	θ_{i11}	$\cdots$	θ_{ij1}	$\cdots$	θ_{iq_i1}
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
a_{ik}	θ_{i1k}	$\cdots$	θ_{ijk}	$\cdots$	θ_{iq_ik}
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
a_{ir_i}	θ_{i1r_i}	$\cdots$	θ_{ijr_i}	$\cdots$	$\theta_{iq_ir_i}$

$$P(A_i = a_{ik} \mid \text{parents}(A_i) = Q_{ij}) = \theta_{ijk}$$

$$\sum_{k=1}^{r_i} \theta_{ijk} = 1$$

(A) Test whether a distribution is decomposable w. r. t. a given graph.

This is the most direct approach. It is not bound to a graphical representation, but can also be carried out w.r.t. other representations of the set of subspaces to be used to compute the (candidate) decomposition of the given distribution.

(B) Find a suitable graph by measuring the strength of dependences.

This is a heuristic, but often highly successful approach, which is based on the frequently valid assumption that in a conditional independence graph an attribute is more strongly dependent on adjacent attributes than on attributes that are not directly connected to them.

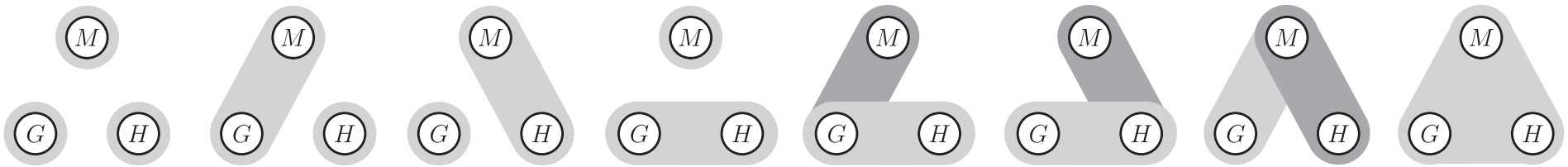
(C) Find an independence map by conditional independence tests.

This approach exploits the theorems that connect conditional independence graphs and graphs that represent decompositions. It has the advantage that a single conditional independence test, if it fails, can exclude several candidate graphs. However, wrong test results can thus have severe consequences.

Evaluation Measures and Search Methods

- All learning algorithms for graphical models consist of
  an **evaluation measure** or **scoring function**
  and a (heuristic) **search method**, e. g.
  - conditional independence search
  - greedy search (spanning tree or K2 algorithm)
  - guided random search (simulated annealing, genetic algorithms)

- An exhaustive search over all graphs is too expensive:
  - $2^{\binom{n}{2}}$ possible undirected graphs for n attributes.
  - $f(n) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} 2^{i(n-i)} f(n-i)$ possible directed acyclic graphs.



8 possible undirected graphs with 3 nodes

Evaluation Measures / Scoring Functions

Relational Networks

- Hartley Information Gain
- Conditional Hartley Information Gain

Probabilistic Networks

- χ^2-Measure
- Mutual Information / Cross Entropy / Information Gain
- (Symmetric) Information Gain Ratio
- (Symmetric/Modified) Gini Index
- Bayesian Measures (K2 metric, BDeu metric)
- Measures based on the Minimum Description Length Principle
- Other measures that are known from Decision Tree Induction

Learning the Structure of Graphical Models from Data

(A) Test whether a distribution is decomposable w. r. t. a given graph.

This is the most direct approach. It is not bound to a graphical representation, but can also be carried out w.r.t. other representations of the set of subspaces to be used to compute the (candidate) decomposition of the given distribution.

(B) Find a suitable graph by measuring the strength of dependences.

This is a heuristic, but often highly successful approach, which is based on the frequently valid assumption that in a conditional independence graph an attribute is more strongly dependent on adjacent attributes than on attributes that are not directly connected to them.

(C) Find an independence map by conditional independence tests.

This approach exploits the theorems that connect conditional independence graphs and graphs that represent decompositions. It has the advantage that a single conditional independence test, if it fails, can exclude several candidate graphs. However, wrong test results can thus have severe consequences.

Testing for Decomposability: Comparing Relations

- In order to evaluate a graph structure, we need a measure that compares the actual relation to the relation represented by the graph.
- For arbitrary R, E_1, and E_2 it is

$$R(E_1 \cap E_2) \leq \min\{R(E_1), R(E_2)\}.$$

- This relation entails that for any family $\mathcal{M}$ of subsets of U it is always:

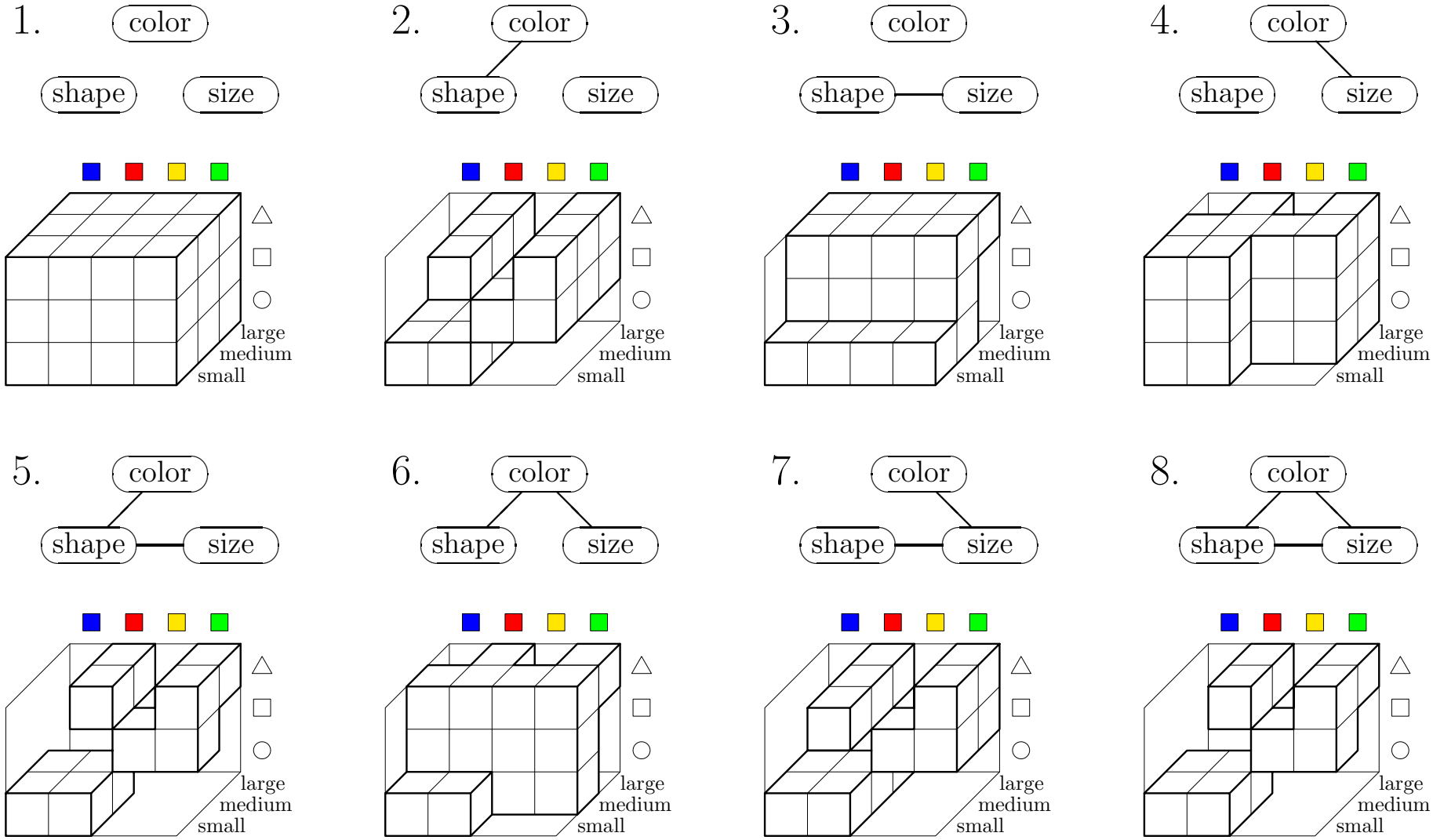
$$\forall a_1 \in \mathrm{dom}(A_1) : \ldots \forall a_n \in \mathrm{dom}(A_n) :$$
$$r_U\left(\bigwedge_{A_i \in U} A_i = a_i\right) \leq \min_{M \in \mathcal{M}} \left\{ r_M\left(\bigwedge_{A_i \in M} A_i = a_i\right)\right\}.$$

- Therefore: Measure the quality of a family $\mathcal{M}$ as:

$$\sum_{a_1 \in \mathrm{dom}(A_1)} \cdots \sum_{a_n \in \mathrm{dom}(A_n)} \left(\min_{M \in \mathcal{M}} \left\{ r_M\left(\bigwedge_{A_i \in M} A_i = a_i\right)\right\} - r_U\left(\bigwedge_{A_i \in U} A_i = a_i\right)\right)$$

Intuitively: **Count the number of additional tuples.**

Direct Test for Decomposability: Relational



Comparing Probability Distributions

Definition: Let P_1 and P_2 be two strictly positive probability distributions on the same set $\mathcal{E}$ of events. Then

$$I_{\mathrm{KLdiv}}(P_1, P_2) = \sum_{F \in \mathcal{E}} P_1(F) \log_2 \frac{P_1(F)}{P_2(F)}$$

is called the **Kullback-Leibler information divergence** of P_1 and P_2.

- The Kullback-Leibler information divergence is non-negative.
- It is zero if and only if $P_1 \equiv P_2$.
- Therefore it is plausible that this measure can be used to assess the quality of the approximation of a given multi-dimensional distribution P_1 by the distribution P_2 that is represented by a given graph:

  The smaller the value of this measure, the better the approximation.

Excursus: Shannon Entropy

Let X be a random variable with domain $\mathrm{dom}(X) = \{x_1, \ldots, x_n\}$. Then,

$$H^{(\text{Shannon})}(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$$

is called the **Shannon entropy** of (the probability distribution of) X,
where $0 \cdot \log_2 0 = 0$ is assumed.

Intuitively: **Expected number of yes/no questions that have to be asked in order to determine the obtaining value of X.**

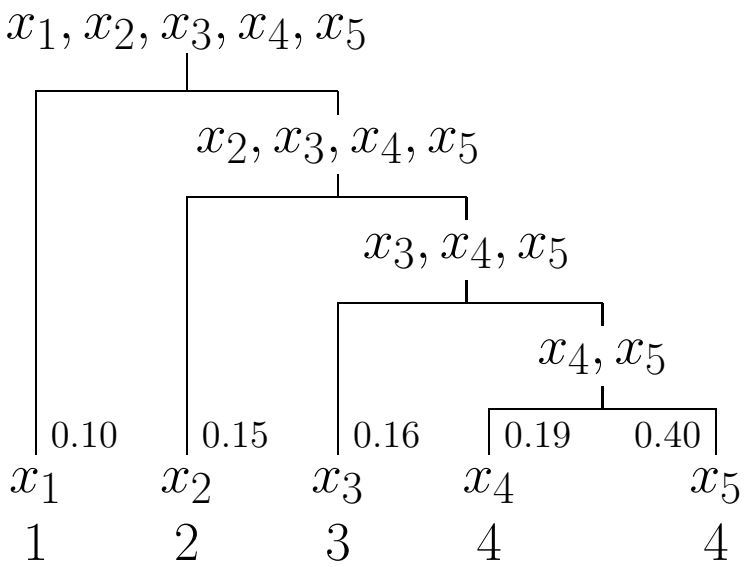
- Suppose there is an oracle, which knows the obtaining value, but responds only if the question can be answered with "yes" or "no".
- A better question scheme than asking for one alternative after the other can easily be found: Divide the set into two subsets of about equal size.
- Ask for containment in an arbitrarily chosen subset.
- Apply this scheme recursively $\rightarrow$ number of questions bounded by $\lceil \log_2 n \rceil$.

Question/Coding Schemes

$$P(x_1) = 0.10, \quad P(x_2) = 0.15, \quad P(x_3) = 0.16, \quad P(x_4) = 0.19, \quad P(x_5) = 0.40$$

Shannon entropy: $-\sum_i P(x_i) \log_2 P(x_i) = 2.15$ bit/symbol

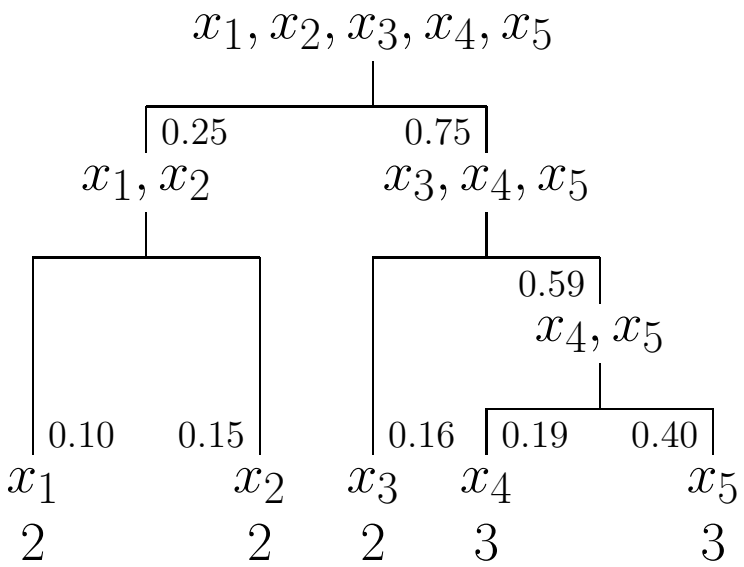
Linear Traversal



Code length: 3.24 bit/symbol

Code efficiency: 0.664

Equal Size Subsets



Code length: 2.59 bit/symbol

Code efficiency: 0.830

Question/Coding Schemes

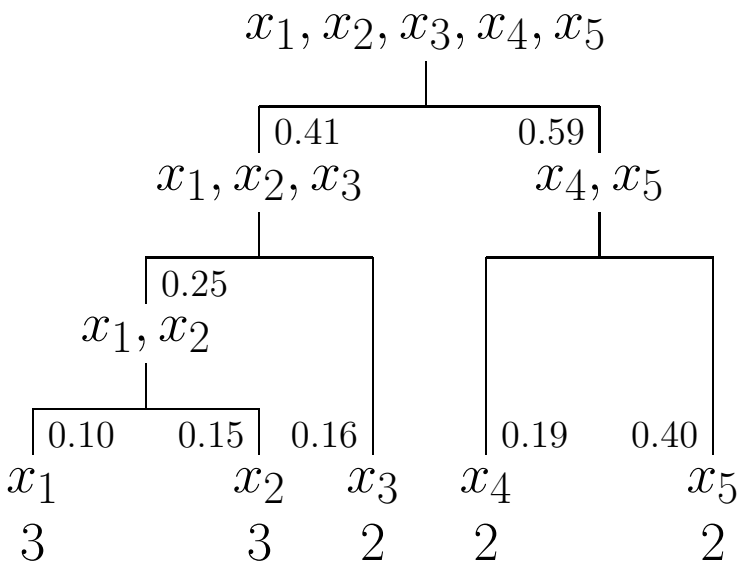
- Splitting into subsets of about equal size can lead to a bad arrangement of the alternatives into subsets → high expected number of questions.
- Good question schemes take the probability of the alternatives into account.
- **Shannon-Fano Coding** (1948)
  - Build the question/coding scheme top-down.
  - Sort the alternatives w.r.t. their probabilities.
  - Split the set so that the subsets have about equal *probability* (splits must respect the probability order of the alternatives).
- **Huffman Coding** (1952)
  - Build the question/coding scheme bottom-up.
  - Start with one element sets.
  - Always combine those two sets that have the smallest probabilities.

Question/Coding Schemes

$$P(x_1) = 0.10, \quad P(x_2) = 0.15, \quad P(x_3) = 0.16, \quad P(x_4) = 0.19, \quad P(x_5) = 0.40$$

Shannon entropy: $-\sum_i P(x_i) \log_2 P(x_i) = 2.15$ bit/symbol

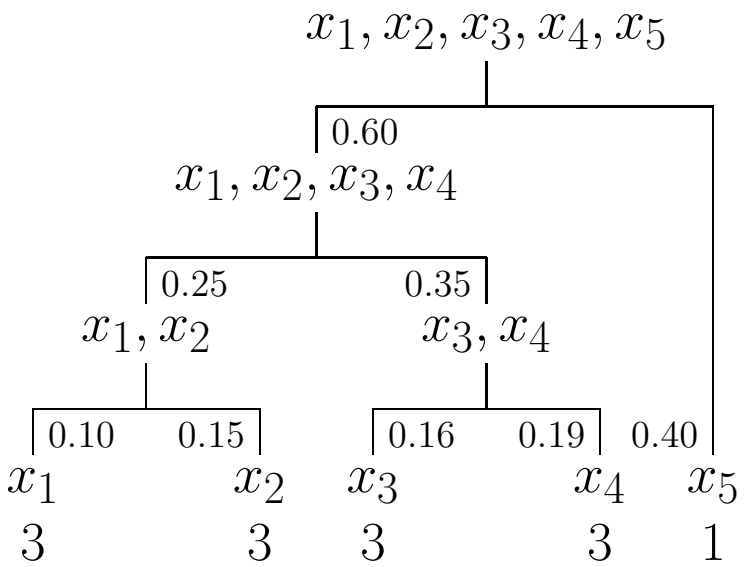
Shannon–Fano Coding (1948)



Code length: 2.25 bit/symbol

Code efficiency: 0.955

Huffman Coding (1952)



Code length: 2.20 bit/symbol

Code efficiency: 0.977

Question/Coding Schemes

- It can be shown that Huffman coding is optimal if we have to determine the obtaining alternative in a single instance.
  (No question/coding scheme has a smaller expected number of questions.)
- Only if the obtaining alternative has to be determined in a sequence of (independent) situations, this scheme can be improved upon.
- Idea: Process the sequence not instance by instance, but combine two, three or more consecutive instances and ask directly for the obtaining combination of alternatives.
- Although this enlarges the question/coding scheme, the expected number of questions per identification is reduced (because each interrogation identifies the obtaining alternative for several situations).
- However, the expected number of questions per identification cannot be made arbitrarily small. Shannon showed that there is a lower bound, namely the Shannon entropy.

Interpretation of Shannon Entropy

$$P(x_1) = \tfrac{1}{2}, \quad P(x_2) = \tfrac{1}{4}, \quad P(x_3) = \tfrac{1}{8}, \quad P(x_4) = \tfrac{1}{16}, \quad P(x_5) = \tfrac{1}{16}$$

Shannon entropy: $-\sum_i P(x_i) \log_2 P(x_i) = 1.875$ bit/symbol

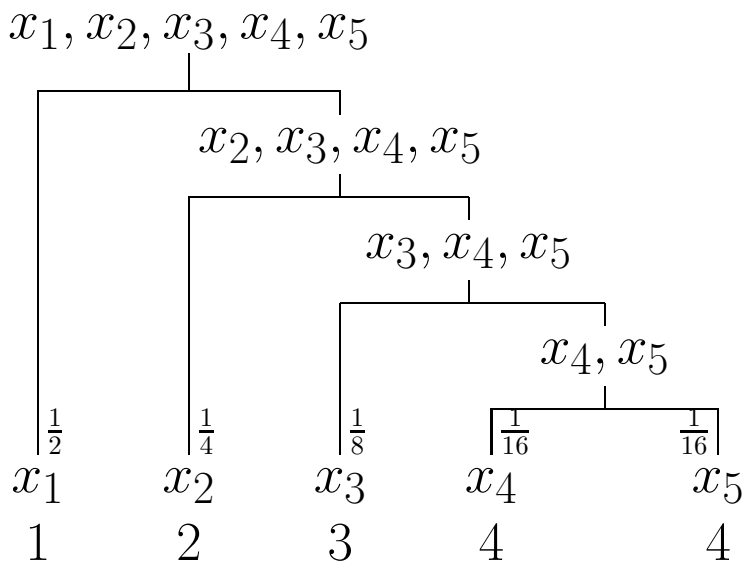
If the probability distribution allows for a perfect Huffman code (code efficiency 1), the Shannon entropy can easily be interpreted as follows:

$$-\sum_i P(x_i) \log_2 P(x_i)$$

$$= \sum_i \underbrace{P(x_i)}_{\substack{\text{occurrence} \\ \text{probability}}} \cdot \underbrace{\log_2 \frac{1}{P(x_i)}}_{\substack{\text{path length} \\ \text{in tree}}} .$$

In other words, it is the expected number of needed yes/no questions.

Perfect Question Scheme



Code length: 1.875 bit/symbol
Code efficiency: 1

Information Content

The information content of an event $F \in \mathcal{E}$ that occurs with probability $P(F)$ is defined as

$$\mathrm{Inf}_P(F) = -\log_2 P(F).$$

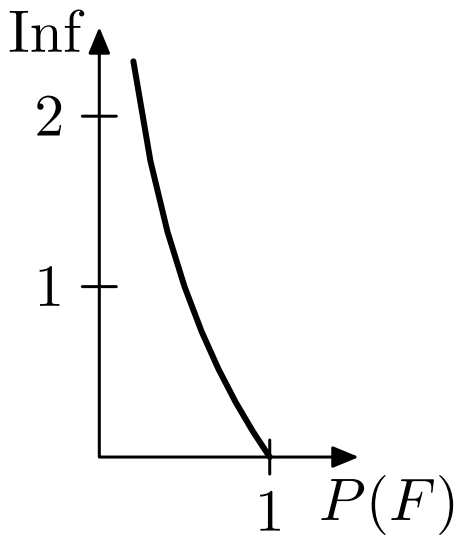
Intention:

- Neglect all subjective references to F and let the information content be determined by $P(F)$ only.
- The information of a certain message ($P(\Omega) = 1$) is zero.
- The less frequent a message occurs (i. e., the less probable it is), the more interesting is the fact of its occurrence:

$$P(F_1) < P(F_2) \quad \Rightarrow \quad \mathrm{Inf}_P(F_1) > \mathrm{Inf}_P(F_2)$$

- We only use one bit to encode the occurrence of a message with probability $\frac{1}{2}$.

The function Inf fulfills all these requirements:



- The expected value (w. r. t. to a probability distribution P_1) of Inf_{P_2} can be written as follows:

$$E_{P_1}(\mathrm{Inf}_{P_2}) = -\sum_{F \in \mathcal{E}} P_1(F) \cdot \log_2 P_2(F)$$

- $H^{(\mathrm{Shannon})}(P)$ is the expected value (in bits) of the information content that is related to the occurrence of the events $F \in \mathcal{E}$:

$$H(P) = E_P(\mathrm{Inf}_P)$$

$$H^{(\mathrm{Shannon})}(P) = \sum_{F \in \mathcal{E}} \underbrace{P(F)}_{\text{Probability of } F} \cdot \underbrace{(-\log_2 P(F))}_{\text{Information content of } F}$$

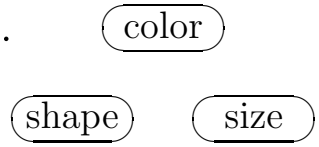
Excursus: Approximation Measure

- Let P^* be a hypothetical probability distribution and P a (given or known) probability distribution that acts as a reference.

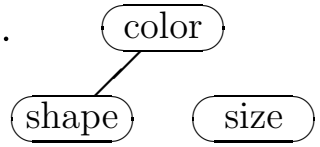
- We can compare both P^* and P by computing the **difference of the expected information contents**:

$$\begin{aligned}
E_P(\mathrm{Inf}_{P^*}) - E_P(\mathrm{Inf}_P) &= -\sum_{F \in \mathcal{E}} P(F) \log_2 P^*(F) + \sum_{F \in \mathcal{E}} P(F) \log_2 P(F) \\
&= \sum_{F \in \mathcal{E}} \Big(P(F) \log_2 P(F) - P(F) \log_2 P^*(F) \Big) \\
&= \sum_{F \in \mathcal{E}} P(F) \Big(\log_2 P(F) - \log_2 P^*(F) \Big) \\
I_{\mathrm{KLdiv}}(P, P^*) &= \sum_{F \in \mathcal{E}} P(F) \log_2 \frac{P(F)}{P^*(F)}
\end{aligned}$$

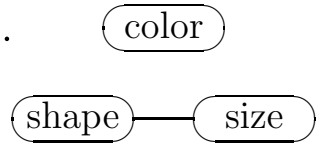
Direct Test for Decomposability: Probabilistic

1. 

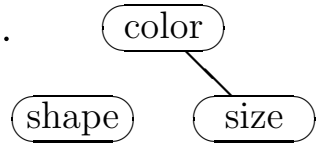
0.640
−5041

2. 

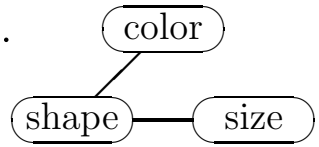
0.211
−4612

3. 

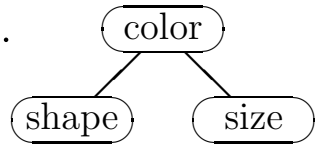
0.429
−4830

4. 

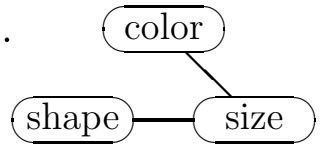
0.590
−4991

5. 

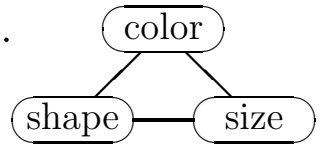
0
−4401

6. 

0.161
−4563

7. 

0.379
−4780

8. 

0
−4401

Upper numbers: The Kullback-Leibler information divergence of the original distribution and its approximation.

Lower numbers: The binary logarithms of the probability of an example database (log-likelihood of data).

Learning the Structure of Graphical Models from Data

(A) Test whether a distribution is decomposable w. r. t. a given graph.

This is the most direct approach. It is not bound to a graphical representation, but can also be carried out w.r.t. other representations of the set of subspaces to be used to compute the (candidate) decomposition of the given distribution.

(B) Find a suitable graph by measuring the strength of dependences.

This is a heuristic, but often highly successful approach, which is based on the frequently valid assumption that in a conditional independence graph an attribute is more strongly dependent on adjacent attributes than on attributes that are not directly connected to them.

(C) Find an independence map by conditional independence tests.

This approach exploits the theorems that connect conditional independence graphs and graphs that represent decompositions. It has the advantage that a single conditional independence test, if it fails, can exclude several candidate graphs. However, wrong test results can thus have severe consequences.

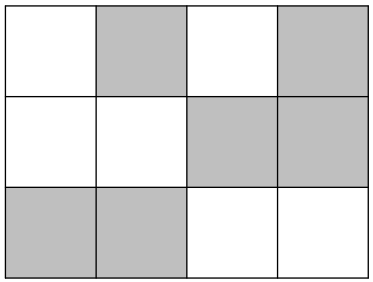
Strength of Marginal Dependences: Relational

- Learning a relational network consists in finding those subspace, for which the intersection of the cylindrical extensions of the projections to these subspaces approximates best the set of possible world states, i. e. contains as few additional tuples as possible.
- Since computing explicitly the intersection of the cylindrical extensions of the projections and comparing it to the original relation is too expensive, local evaluation functions are used, for instance:

subspace	color × shape	shape × size	size × color
possible combinations	12	9	12
occurring combinations	6	5	8
relative number	50%	56%	67%

- The relational network can be obtained by interpreting the relative numbers as edge weights and constructing the minimum weight spanning tree.

Strength of Marginal Dependences: Relational



Hartley information needed to determine

coordinates:	$\log_2 4 + \log_2 3 = \log_2 12$	≈ 3.58
coordinate pair:	$\log_2 6$	≈ 2.58
gain:	$\log_2 12 - \log_2 6 = \log_2 2 = 1$	

Definition: Let A and B be two attributes and R a discrete possibility measure with $\exists a \in \mathrm{dom}(A) : \exists b \in \mathrm{dom}(B) : R(A = a, B = b) = 1$. Then

$$
\begin{aligned}
I_{\mathrm{gain}}^{(\mathrm{Hartley})}(A, B) &= \log_2 \left(\textstyle\sum_{a \in \mathrm{dom}(A)} R(A = a) \right) + \log_2 \left(\textstyle\sum_{b \in \mathrm{dom}(B)} R(B = b) \right) \\
&- \log_2 \left(\textstyle\sum_{a \in \mathrm{dom}(A)} \sum_{b \in \mathrm{dom}(B)} R(A = a, B = b) \right) \\
&= \log_2 \frac{\left(\sum_{a \in \mathrm{dom}(A)} R(A = a) \right) \cdot \left(\sum_{b \in \mathrm{dom}(B)} R(B = b) \right)}{\sum_{a \in \mathrm{dom}(A)} \sum_{b \in \mathrm{dom}(B)} R(A = a, B = b)},
\end{aligned}
$$

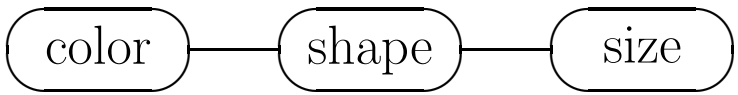
is called the **Hartley information gain** of A and B w.r.t. R.

Strength of Marginal Dependences: Simple Example

- **Intuitive interpretation of Hartley information gain:**
  The binary logarithm measures the number of questions to find the obtaining value with a scheme like a binary search. Thus Hartley information gain measures the reduction in the number of necessary questions.
- Results for the simple example:

$$\begin{aligned} I_{\text{gain}}^{(\text{Hartley})}(\text{color}, \text{shape}) &= 1.00 \text{ bit} \\ I_{\text{gain}}^{(\text{Hartley})}(\text{shape}, \text{size}) &\approx 0.86 \text{ bit} \\ I_{\text{gain}}^{(\text{Hartley})}(\text{color}, \text{size}) &\approx 0.58 \text{ bit} \end{aligned}$$

- Applying the Kruskal algorithm yields as a learning result:



  As we know, this graph describes indeed a decomposition of the relation.

Mutual Information / Cross Entropy / Information Gain

Based on Shannon Entropy $H = -\sum_{i=1}^{n} p_i \log_2 p_i$ (Shannon 1948)

$$
\begin{aligned}
I_{\text{gain}}(A, B) &= \overbrace{H(A)}^{} - \overbrace{H(A \mid B)}^{} \\
&= \overbrace{-\sum_{\forall a} P(a) \log_2 P(a)}^{} - \overbrace{\sum_{\forall b} P(b) \left(-\sum_{\forall a} P(a|b) \log_2 P(a|b) \right)}^{}
\end{aligned}
$$

$H(A)$	Entropy of the distribution on attribute A
$H(A\|B)$	*Expected entropy* of the distribution on attribute A if the value of attribute B becomes known
$H(A) - H(A\|B)$	Expected reduction in entropy or *information gain*

Strength of Marginal Dependences: Probabilistic

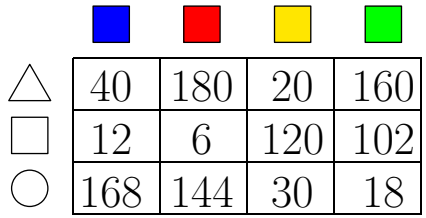
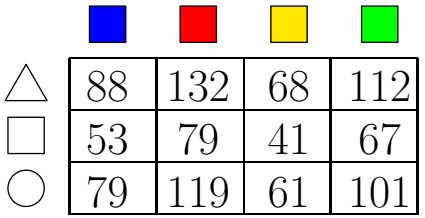
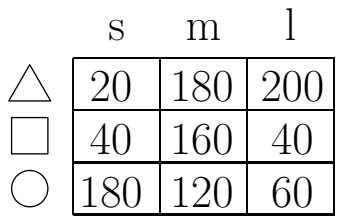
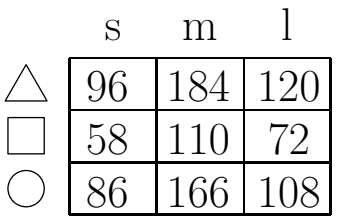
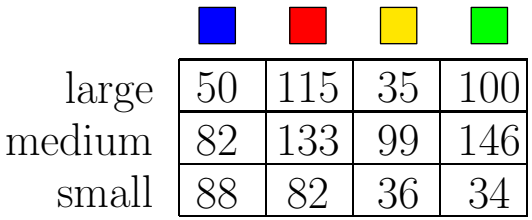
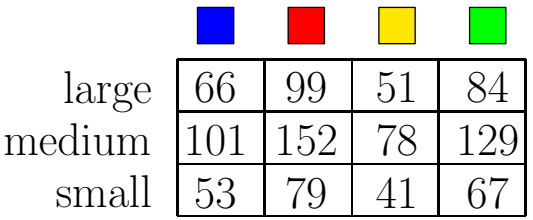
$$
\begin{aligned}
I_{\text{gain}}(A,B) &= -\sum_{\forall a} P(a)\log_2 P(a) - \sum_{\forall b} P(b)\left(-\sum_{\forall a} P(a|b)\log_2 P(a|b)\right) \\
&= -\sum_{\forall a}\sum_{\forall b} P(a,b)\log_2 P(a) + \sum_{\forall b}\sum_{\forall a} P(a|b)P(b)\log_2 P(a|b) \\
&= \sum_{\forall a}\sum_{\forall b} P(a,b)\Big(\log_2 \frac{P(a,b)}{P(b)} - \log_2 P(a)\Big) \\
&= \sum_{\forall a}\sum_{\forall b} P(a,b)\log_2 \frac{P(a,b)}{P(a)P(b)}
\end{aligned}
$$

The information gain equals the Kullback-Leibler information divergence between the actual distribution $P(A,B)$ and a hypothetical distribution P^* in which A and B are marginal independent:

$$P^*(A,B) = P(A)\cdot P(B)$$

$$I_{\text{gain}}(A,B) = I_{\text{KLdiv}}(P,P^*)$$

Information Gain: Simple Example

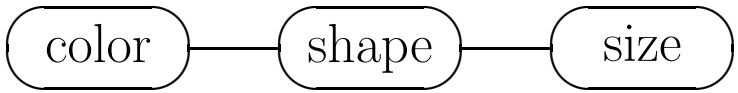
	projection to subspace	product of marginals	information gain
			0.429 bit
			0.211 bit
			0.050 bit

Strength of Marginal Dependences: Simple Example

- Results for the simple example:

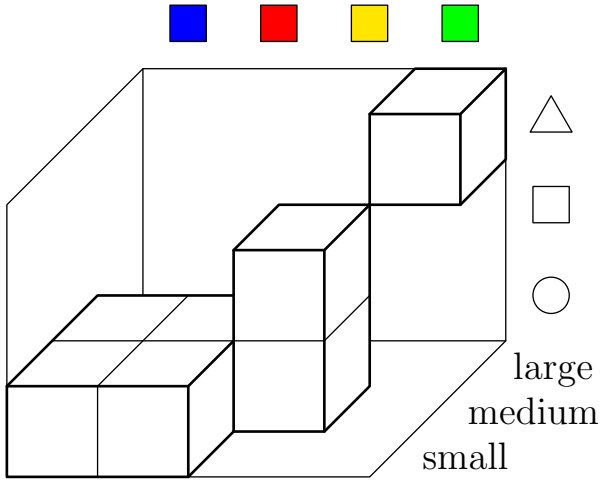
$$\begin{aligned} I_{\text{gain}}(\text{color}, \text{shape}) &= 0.429 \text{ bit} \\ I_{\text{gain}}(\text{shape}, \text{size}) &= 0.211 \text{ bit} \\ I_{\text{gain}}(\text{color}, \text{size}) &= 0.050 \text{ bit} \end{aligned}$$

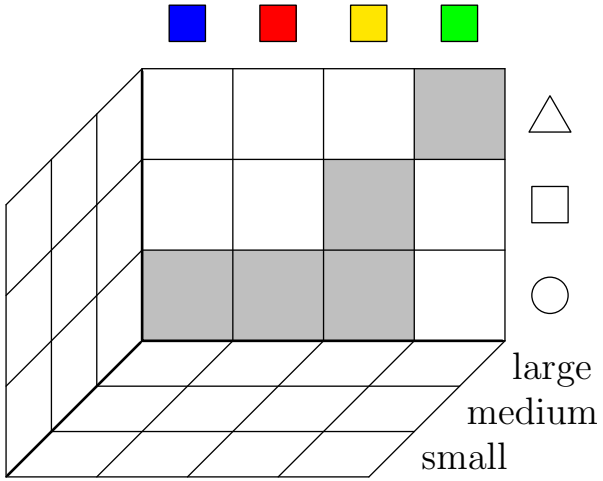
- Applying the Kruskal algorithm yields as a learning result:

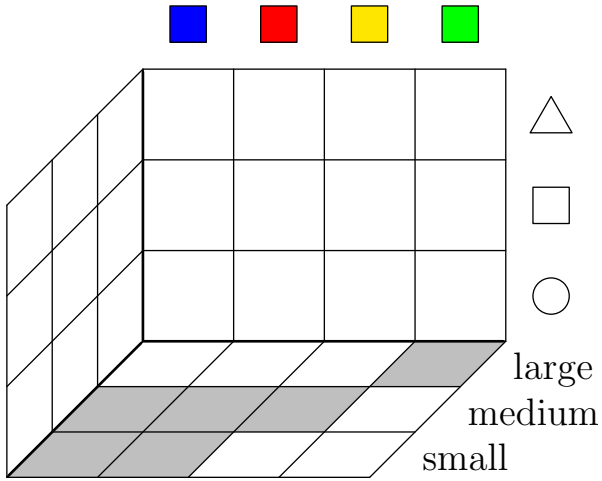


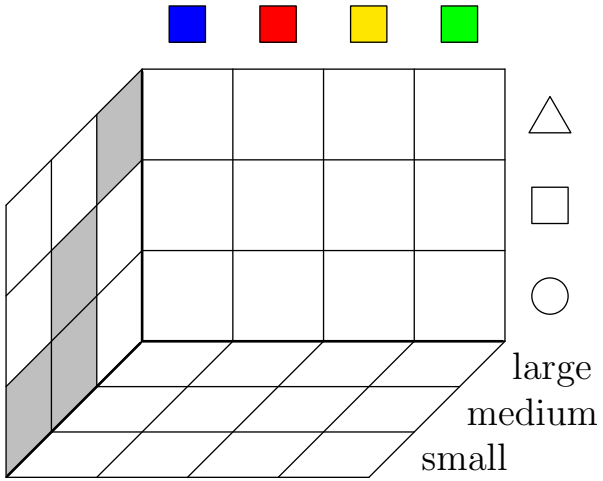
- It can be shown that this approach always yields the best possible spanning tree w.r.t. Kullback-Leibler information divergence (Chow and Liu 1968).
- In an extended form this also holds for certain classes of graphs (for example, tree-augmented naive Bayes classifiers).
- For more complex graphs, the best graph need not be found (there are counterexamples, see below).

Strength of Marginal Dependences: Drawbacks

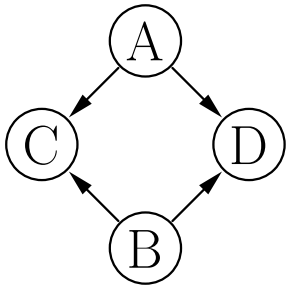








Strength of Marginal Dependences: Drawbacks



p_A	a_1	a_2
	0.5	0.5

p_B	b_1	b_2
	0.5	0.5

$p_{C\|AB}$	a_1b_1	a_1b_2	a_2b_1	a_2b_2
c_1	0.9	0.3	0.3	0.5
c_2	0.1	0.7	0.7	0.5

$p_{D\|AB}$	a_1b_1	a_1b_2	a_2b_1	a_2b_2
d_1	0.9	0.3	0.3	0.5
d_2	0.1	0.7	0.7	0.5

p_{AD}	a_1	a_2
d_1	0.3	0.2
d_2	0.2	0.3

p_{BD}	b_1	b_2
d_1	0.3	0.2
d_2	0.2	0.3

p_{CD}	c_1	c_2
d_1	0.31	0.19
d_2	0.19	0.31

- Greedy parent selection can lead to suboptimal results if there is more than one path connecting two attributes.
- Here: the edge $C \to D$ is selected first.

Strength of Marginal Dependences: General Algorithms

- **Optimum Weight Spanning Tree Construction**
  - Compute an evaluation measure on all possible edges (two-dimensional subspaces).
  - Use the Kruskal algorithm to determine an optimum weight spanning tree.

- **Greedy Parent Selection** (for directed graphs)
  - Define a topological order of the attributes (to restrict the search space).
  - Compute an evaluation measure on all single attribute hyperedges.
  - For each preceding attribute (w.r.t. the topological order): add it as a candidate parent to the hyperedge and compute the evaluation measure again.
  - Greedily select a parent according to the evaluation measure.
  - Repeat the previous two steps until no improvement results from them.

K2 Algorithm

- Idea: Compute the probability of a directed graph $\vec{G}$ given the database D (Bayesian approach by [Cooper and Herskovits 1992])

$$\vec{G}_{\text{opt}} = \arg\max_{\vec{G}} P(\vec{G} \mid D) = \arg\max_{\vec{G}} \frac{P(\vec{G}, D)}{P(D)}$$

$$= \arg\max_{\vec{G}} P(\vec{G}, D)$$

$\Rightarrow$ Find an equation for $P(\vec{G}, D)$.

- In order to compare two graphs, it is sufficient to compute the **Bayes factor**

$$\frac{P(\vec{G}_1 \mid D)}{P(\vec{G}_2 \mid D)} = \frac{P(\vec{G}_1, D)}{P(\vec{G}_2, D)}$$

In both ways one can avoid computing the probability $P(D)$.
Assuming equal probability of all graphs simplifies further.

Model Averaging

We first consider $P(\vec{G}, D)$ to be the marginalization of $P(\vec{G}, \Theta, D)$ over all possible parameters Θ.

$$
\begin{aligned}
P(\vec{G}, D) &= \int_\Theta P(\vec{G}, \Theta, D)\,\mathrm{d}\Theta \\
&= \int_\Theta P(D \mid \vec{G}, \Theta)\, P(\vec{G}, \Theta)\,\mathrm{d}\Theta \\
&= \int_\Theta P(D \mid \vec{G}, \Theta)\, f(\Theta \mid \vec{G}) P(\vec{G})\,\mathrm{d}\Theta \\
&= \underbrace{P(\vec{G})}_{\text{A priori prob.}} \int_\Theta \underbrace{P(D \mid \vec{G}, \Theta)}_{\text{Likelihood of } D} \underbrace{f(\Theta \mid \vec{G})}_{\text{Parameter densities}} \mathrm{d}\Theta
\end{aligned}
$$

K2 Algorithm

- The a priori distribution $P(\vec{G})$ can be used to bias the evaluation measure towards user-specific network structures.
- Substitute the likelihood $P(D \mid \vec{G}, \Theta)$ for its specific form:

$$P(\vec{G}, D) \;=\; P(\vec{G}) \int_{\Theta} \underbrace{\left[\prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}} \right]}_{P(D|\vec{G},\Theta)} f(\Theta \mid \vec{G}) \, \mathrm{d}\Theta$$

- See slide 300 for the derivation of the likelihood term.

K2 Algorithm

- The parameter densities $f(\Theta \mid \vec{G})$ describe the probabilities of the parameters given a network structure.
- They are densities of second order (distribution over distributions)
- For fixed i and j, a vector $(\theta_{ij1}, \ldots, \theta_{ijr_i})$ represents a probability distribution, namely the j-th column of the i-th potential table.
- Assuming mutual independence between the potential tables, we arrive for $f(\Theta \mid \vec{G})$ at the following:

$$f(\Theta \mid \vec{G}) \quad = \quad \prod_{i=1}^{n} \prod_{j=1}^{q_i} f(\theta_{ij1}, \ldots, \theta_{ijr_i})$$

K2 Algorithm

- Thus, we can further concretize the equation for $P(\vec{G}, D)$:

$$
\begin{aligned}
P(\vec{G}, D) &= P(\vec{G}) \int_{\theta_{ijk}} \cdots \int \left[\prod_{i=1}^{n} \prod_{j=1}^{q_i} \prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}} \right] \cdot \left[\prod_{i=1}^{n} \prod_{j=1}^{q_i} f(\theta_{ij1}, \ldots, \theta_{ijr_i}) \right] \, \mathrm{d}\theta_{111}, \ldots, \mathrm{d}\theta_{nq_nr_n} \\
&= P(\vec{G}) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \int_{\theta_{ijk}} \cdots \int \left[\prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}} \right] \cdot f(\theta_{ij1}, \ldots, \theta_{ijr_i}) \, \mathrm{d}\theta_{ij1}, \ldots, \mathrm{d}\theta_{ijr_i}
\end{aligned}
$$

K2 Algorithm

- A last assumption: For fixed i and j the density $f(\theta_{ij1}, \ldots, \theta_{ijr_i})$ is uniform:

$$f(\theta_{ij1}, \ldots, \theta_{ijr_i}) \;=\; (r_i - 1)!$$

- It simplifies $P(\vec{G}, D)$ further:

$$\begin{aligned} P(\vec{G}, D) &= P(\vec{G}) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \int_{\theta_{ijk}} \cdots \int \left[\prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}} \right] \cdot (r_i - 1)! \, \mathrm{d}\theta_{ij1}, \ldots, \mathrm{d}\theta_{ijr_i} \\ &= P(\vec{G}) \prod_{i=1}^{n} \prod_{j=1}^{q_i} (r_i - 1)! \underbrace{\int_{\theta_{ijk}} \cdots \int \prod_{k=1}^{r_i} \theta_{ijk}^{\alpha_{ijk}} \, \mathrm{d}\theta_{ij1}, \ldots, \mathrm{d}\theta_{ijr_i}}_{\text{Dirichlet's integral} = \frac{\prod_{k=1}^{r_i} \alpha_{ijk}!}{(\sum_{k=1}^{r_i} \alpha_{ijk} + r_i - 1)!}} \end{aligned}$$

K2 Algorithm

- We finally arrive at an expression for $P(\vec{G}, D)$:

$$P(\vec{G}, D) \quad = \quad \mathrm{K2}(\vec{G} \mid D) \quad = \quad P(\vec{G}) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \left[\frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} \alpha_{ijk}! \right]$$

n	number of attributes describing the domain under consideration
r_i	number of values of the i-th attribute A_i, i. e., $r_i = \lvert\mathrm{dom}(A_i)\rvert$
q_i	number of instantiations of the parents of the i-th attribute in $\vec{G}$, i. e., $q_i = \prod_{A_j \in \mathrm{parents}(A_i)} r_i = \prod_{A_j \in \mathrm{parents}(A_i)} \lvert\mathrm{dom}(A_i)\rvert$
α_{ijk}	number of sample cases in which the i-th attribute has its k-th value and its parents in $\vec{G}$ have their j-th instantiation
N_{ij}	$= \sum_{k=1}^{r_i} \alpha_{ijk}$

Properties of the K2 Metric

- **Global** — Refers to the outer product: The total value of the K2 metric is the product over all K2 values of attribute families.
- **Local** — The likelihood equation assumes that given a parents instantiation, the probabilities for the respective child attribute values are mutual independent. This is reflected in the product over all q_i different parent attributes' value combinations of attribute A_i.

We exploit the global property to write the K2 metric as follows:

$$\mathrm{K2}(\vec{G} \mid D) \quad = \quad P(\vec{G}) \prod_{i=1}^{n} \mathrm{K2}_{\mathrm{local}}(A_i \mid D)$$

with

$$\mathrm{K2}_{\mathrm{local}}(A_i \mid D) \quad = \quad \prod_{j=1}^{q_i} \left[\frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} \alpha_{ijk}! \right]$$

K2 Algorithm

Prerequisites:

- Choose a topological order on the attributes $(A_1, \ldots, A_n)$
- Start out with a network that consists of n isolated nodes.
- Let ζ_i be the quality of the i-th attribute given the (tentative) set of parent attributes M:

$$\zeta_i(M) = \mathrm{K2}_{\mathrm{local}}(A_i \mid D) \quad \text{with} \quad \mathrm{parents}(A_i) = M$$

K2 Algorithm

Execution:

1. Determine for the parentless node A_i the quality measure $\zeta_i(\emptyset)$
2. Evaluate for every predecessor $\{A_1, \ldots, A_{i-1}\}$ whether inserted as parent of A_i, the quality measure would increase. Let Y be the node that yields the highest quality (increase):

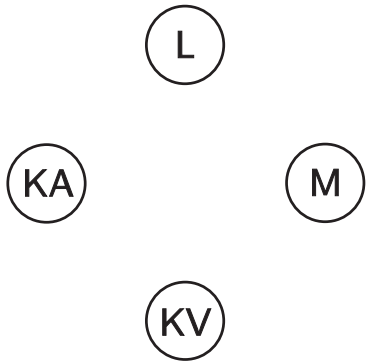
   $$Y = \arg\max_{1 \leq l \leq i-1} \zeta_i(\{A_l\})$$

   This best quality measure be $\zeta = \zeta_i(\{Y\})$.
3. If ζ is better than $\zeta_i(\emptyset)$, Y is inserted permanently as a parent node: $\text{parents}(A_i) = \text{parents}(A_i) \cup \{Y\}$
4. Repeat steps 2 and 3 to increase the parent set until no quality increase can be achieved or no nodes are left or a predefined maximum number of parent nodes per node is reached.

K2 Algorithm

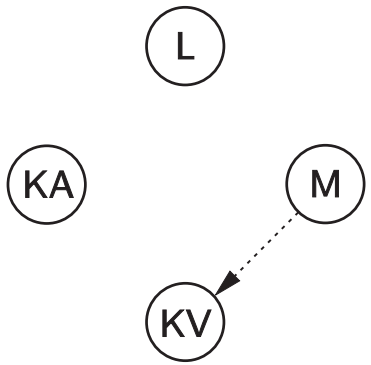
1: **for** $i \leftarrow 1 \ldots n$ **do** // Initialization
2: $\quad \text{parents}(A_i) \leftarrow \emptyset$
3: **end for**
4: **for** $i \leftarrow n, \ldots, 1$ **do** // Iteration
5: $\quad$ **repeat**
6: $\quad\quad$ Select $Y \in \{A_1, \ldots, A_{i-1}\} \setminus \text{parents}(A_i)$,
$\quad\quad\quad$ which maximizes $\zeta = \zeta_i(\text{parents}(A_i) \cup \{Y\})$
7: $\quad\quad \delta \leftarrow \zeta - \zeta_i(\text{parents}(A_i))$
8: $\quad\quad$ **if** $\delta > 0$ **then**
9: $\quad\quad\quad \text{parents}(A_i) \leftarrow \text{parents}(A_i) \cup \{Y\}$
10: $\quad\quad$ **end if**
11: $\quad$ **until** $\delta \leq 0$ or $\text{parents}(A_i) = \{A_1, \ldots, A_{i-1}\}$ or $|\text{parents}(A_i)| = n_{\max}$
12: **end for**

Demo of K2 Algorithm



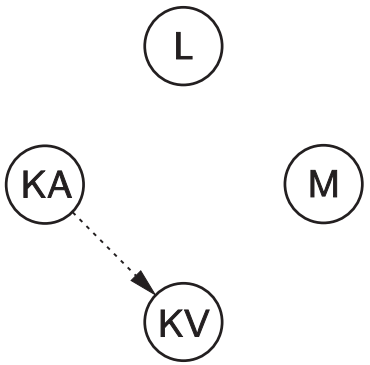
-43942.99

Step 1 – Edgeless graph



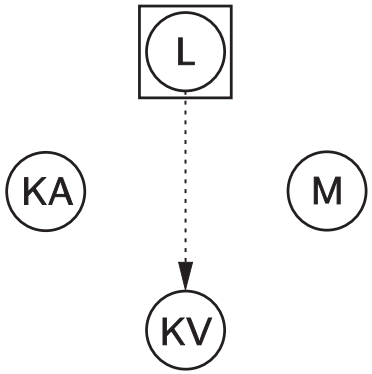
-43950.53

Step 2 – Insert M temporarily.



-43949.30

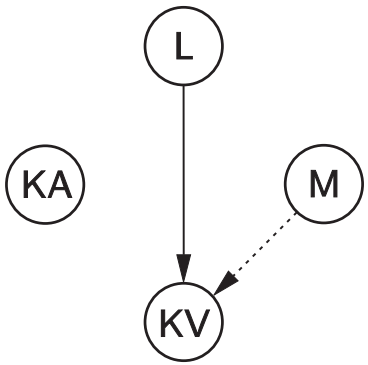
Step 3 – Insert KA temporarily.



-43942.48

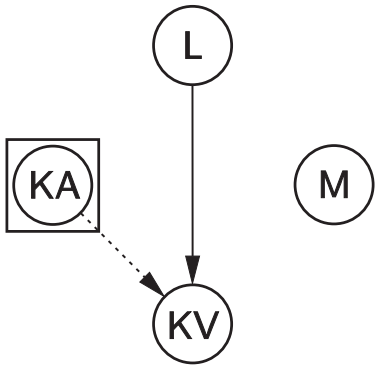
Step 4 – Node L maximizes K2 value and thus is added permantently.

Demo of K2 Algorithm



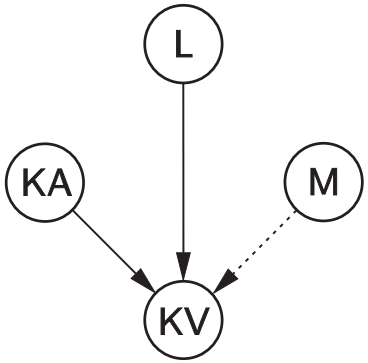
-43964.12

Step 5 – Insert M temporarily.



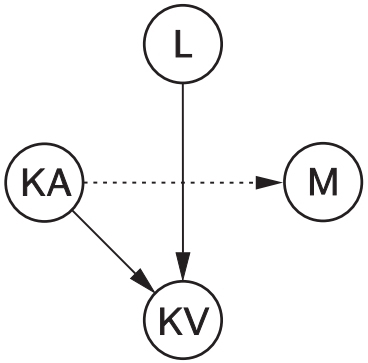
-43933.71

Step 6 – KA is added as second parent node of KV.



-43995.02

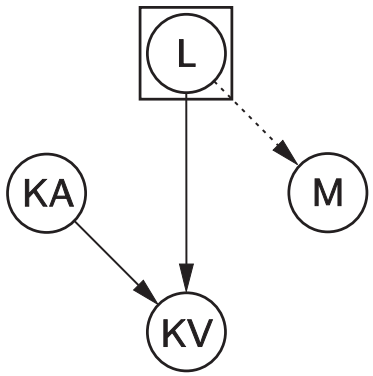
Step 7 – M does not increase the quality of the network if insertes as third parent node.



-43492.08

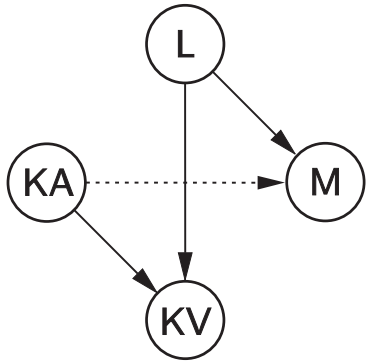
Step 8 – Insert KA temporarily.

Demo of K2 Algorithm



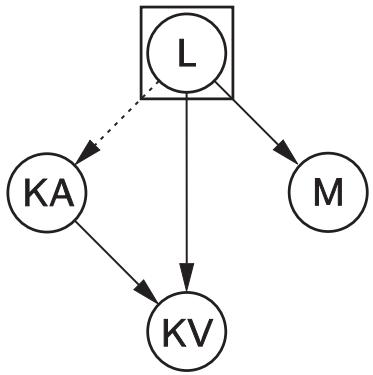
-40823.76

Step 9 – Node L becomes perent node of M.



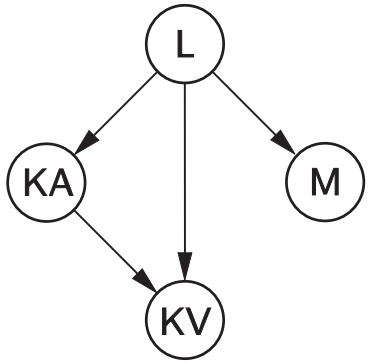
-40903.39

Step 10 – Adding KA does not increase overall network quaility.



-39190.67

Step 11 – Node L becomes parent node of KA.



Result

Learning the Structure of Graphical Models from Data

(A) Test whether a distribution is decomposable w. r. t. a given graph.

This is the most direct approach. It is not bound to a graphical representation, but can also be carried out w.r.t. other representations of the set of subspaces to be used to compute the (candidate) decomposition of the given distribution.

(B) Find a suitable graph by measuring the strength of dependences.

This is a heuristic, but often highly successful approach, which is based on the frequently valid assumption that in a conditional independence graph an attribute is more strongly dependent on adjacent attributes than on attributes that are not directly connected to them.

(C) Find an independence map by conditional independence tests.

This approach exploits the theorems that connect conditional independence graphs and graphs that represent decompositions. It has the advantage that a single conditional independence test, if it fails, can exclude several candidate graphs. However, wrong test results can thus have severe consequences.

Structure Learning with Conditional Independence Tests

General Idea: Exploit the theorems that connect conditional independence graphs and graphs that represent decompositions.

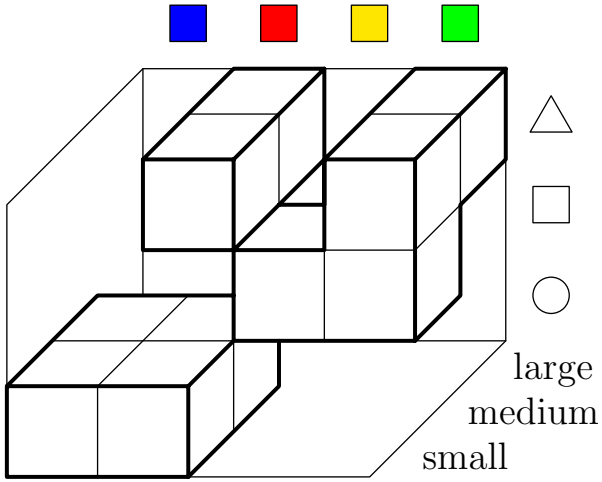
In other words: we want a graph describing a decomposition,
but we search for a conditional independence graph.

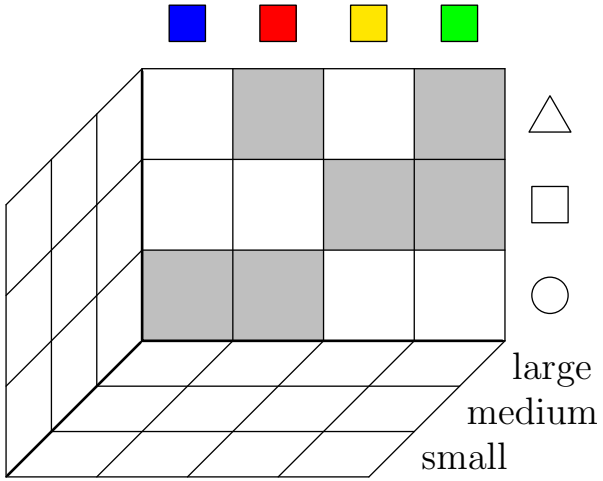
This approach has the advantage that a single conditional independence test, if it fails, can exclude several candidate graphs.

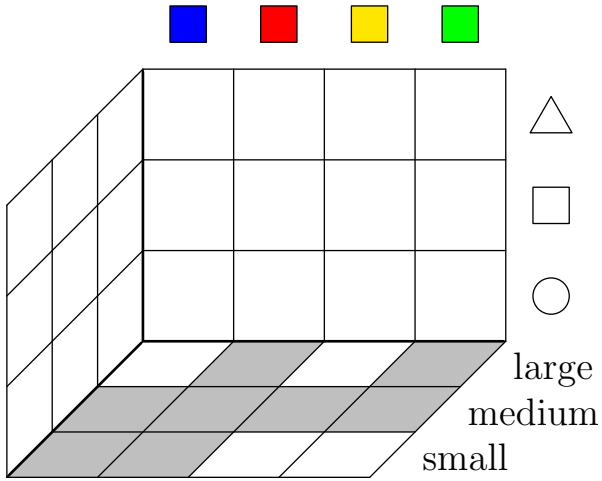
Assumptions:

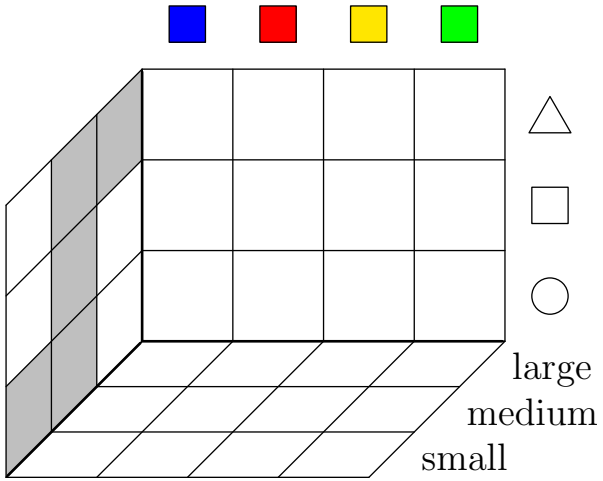
- *Faithfulness:* The domain under consideration can be accurately described with a graphical model (more precisely: there exists a perfect map).
- *Reliability of Tests:* The result of all conditional independence tests coincides with the actual situation in the underlying distribution.
- Other assumptions that are specific to individual algorithms.

Conditional Independence Tests: Relational









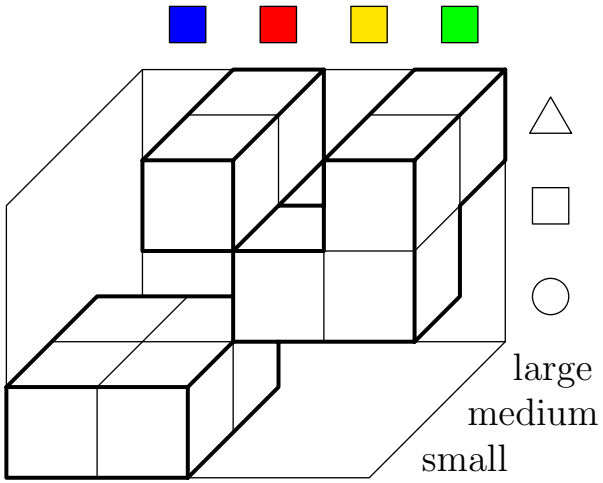
Conditional Independence Tests: Relational

- The Hartley information gain can be used directly to test for (approximate) **marginal independence**.

attributes	relative number of possible value combinations	Hartley information gain
color, shape	$\frac{6}{3 \cdot 4} = \frac{1}{2} = 50\%$	$\log_2 3 + \log_2 4 - \log_2 6 = 1$
color, size	$\frac{8}{3 \cdot 4} = \frac{2}{3} \approx 67\%$	$\log_2 3 + \log_2 4 - \log_2 8 \approx 0.58$
shape, size	$\frac{5}{3 \cdot 3} = \frac{5}{9} \approx 56\%$	$\log_2 3 + \log_2 3 - \log_2 5 \approx 0.85$

- In order to test for (approximate) **conditional independence**:
  - Compute the Hartley information gain for each possible instantiation of the conditioning attributes.
  - Aggregate the result over all possible instantiations, for instance, by simply averaging them.

Conditional Independence Tests: Simple Example



color	Hartley information gain
■	$\log_2 1 + \log_2 2 - \log_2 2 = 0$
■	$\log_2 2 + \log_2 3 - \log_2 4 \approx 0.58$
■	$\log_2 1 + \log_2 1 - \log_2 1 = 0$
■	$\log_2 2 + \log_2 2 - \log_2 2 = 1$
	average: ≈ 0.40

shape	Hartley information gain
△	$\log_2 2 + \log_2 2 - \log_2 4 = 0$
□	$\log_2 2 + \log_2 1 - \log_2 2 = 0$
○	$\log_2 2 + \log_2 2 - \log_2 4 = 0$
	average: $= 0$

size	Hartley information gain
large	$\log_2 2 + \log_2 1 - \log_2 2 = 0$
medium	$\log_2 4 + \log_2 3 - \log_2 6 = 1$
small	$\log_2 2 + \log_2 1 - \log_2 2 = 0$
	average: ≈ 0.33

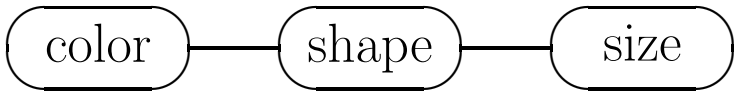
Conditional Independence Tests: Simple Example

- The Shannon information gain can be used directly to test for (approximate) **marginal independence**.
- Conditional independence tests may be carried out by summing the information gain for all instantiations of the conditioning variables:

$$I_{\text{gain}}(A, B \mid C)$$
$$= \sum_{c \in \text{dom}(C)} P(c) \sum_{a \in \text{dom}(A)} \sum_{b \in \text{dom}(B)} P(a, b \mid c) \log_2 \frac{P(a, b \mid c)}{P(a \mid c)\, P(b \mid c)},$$

  where $P(c)$ is an abbreviation of $P(C = c)$ etc.

- Since $I_{\text{gain}}(\text{color}, \text{size} \mid \text{shape}) = 0$ indicates the only conditional independence, we get the following learning result:



Conditional Independence Tests: General Algorithm

Algorithm: (conditional independence graph construction)

1. For each pair of attributes A and B, search for a set $S_{AB} \subseteq U \backslash \{A, B\}$ such that $A \perp\!\!\!\perp B \mid S_{AB}$ holds in $\widehat{P}$, i.e., A and B are independent in $\widehat{P}$ conditioned on S_{AB}. If there is no such S_{AB}, connect the attributes by an undirected edge.

2. For each pair of non-adjacent variables A and B with a common neighbour C (i.e., C is adjacent to A as well as to B), check whether $C \in S_{AB}$.
   - If it is, continue.
   - If it is not, add arrow heads pointing to C, i.e., $A \rightarrow C \leftarrow B$.

3. Recursively direct all undirected edges according to the rules:
   - If for two adjacent variables A and B there is a strictly directed path from A to B not including $A \rightarrow B$, then direct the edge towards B.
   - If there are three variables A, B, and C with A and B not adjacent, $B - C$, and $A \rightarrow C$, then direct the edge $C \rightarrow B$.

Conditional Independence Tests: Simple Example

Suppose that the following conditional independence statements hold:

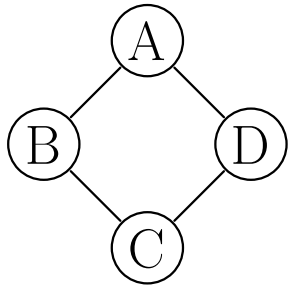
$$
\begin{array}{ll}
A \perp\!\!\!\perp_{\hat{P}} B \mid \emptyset & B \perp\!\!\!\perp_{\hat{P}} A \mid \emptyset \\
A \perp\!\!\!\perp_{\hat{P}} D \mid C & D \perp\!\!\!\perp_{\hat{P}} A \mid C \\
B \perp\!\!\!\perp_{\hat{P}} D \mid C & D \perp\!\!\!\perp_{\hat{P}} B \mid C
\end{array}
$$

All other possible conditional independence statements that can be formed with the attributes A, B, C, and D (with single attributes on the left) do not hold.

- **Step 1:** Since there is no set rendering A and C, B and C and C and D independent, the edges $A - C$, $B - C$, and $C - D$ are inserted.
- **Step 2:** Since C is a common neighbor of A and B and we have $A \perp\!\!\!\perp_{\hat{P}} B \mid \emptyset$, but $A \not\perp\!\!\!\perp_{\hat{P}} B \mid C$, the first two edges must be directed $A \rightarrow C \leftarrow B$.
- **Step 3:** Since A and D are not adjacent, $C - D$ and $A \rightarrow C$, the edge $C - D$ must be directed $C \rightarrow D$.
  (Otherwise step 2 would have already fixed the orientation $C \leftarrow D$.)

Conditional Independence Tests: Drawbacks

- The conditional independence graph construction algorithm presupposes that there is a **perfect map**. If there is no perfect map, the result may be invalid.



p_{ABCD}		$A = a_1$		$A = a_2$	
		$B = b_1$	$B = b_2$	$B = b_1$	$B = b_2$
$C = c_1$	$D = d_1$	$1/47$	$1/47$	$1/47$	$2/47$
	$D = d_2$	$1/47$	$1/47$	$2/47$	$4/47$
$C = c_2$	$D = d_1$	$1/47$	$2/47$	$1/47$	$4/47$
	$D = d_2$	$2/47$	$4/47$	$4/47$	$16/47$

- **Independence tests of high order**, i. e., with a large number of conditions, may be necessary.
- There are approaches to mitigate these drawbacks.
  (For example, the order is restricted and all tests of higher order are assumed to fail, if all tests of lower order failed.)

The Cheng–Bell–Liu Algorithm

- **Drafting:** Build a so-called Chow–Liu tree as an initial graphical model.
  - Evaluate all attribute pairs (candidate edges) with information gain.
  - Discard edges with evaluation below independence threshold ($\sim$0.1 bits).
  - Build optimum (maximum) weight spanning tree.
- **Thickening:** Add necessary edges.
  - Traverse remaining candidate edges in the order of decreasing evaluation.
  - Test for conditional independence in order to determine whether an edge is needed in the graphical model.
  - Use local Markov property to select a condition set: an attribute is conditionally independent of all non-descendants given its parents.
  - Since the graph is undirected in this step, the set of adjacent nodes is reduced iteratively and greedily in order to remove possible children.

The Cheng–Bell–Liu Algorithm (continued)

- **Thinning:** Remove superfluous edges.
  - In the thickening phase a conditional independence test may have failed, because the graph was still too sparse.
  - Traverse all edges that have been added to the current graphical model and test for conditional independence.
  - Remove unnecessary edges.
    (two phases/approaches: heuristic test/strict test)
- **Orienting:** Direct the edges of the graphical model.
  - Identify the v-structures (converging directed edges).
    (Markov equivalence: same skeleton and same set of v-structures.)
  - Traverse all pairs of attributes with common neighbors and check which common neighbors are in the (maximally) reduced set of conditions.
  - Direct remaining edges by extending chains and avoiding cycles.

Learning Undirected Graphical Models Directly

- **Drafting:** Build a Chow–Liu tree as an initial graphical model
  - Evaluate all attribute pairs (candidate edges) with specificity gain.
  - Discard edges with evaluation below independence threshold ($\sim$0.015).
  - Build optimum (maximum) weight spanning tree.
- **Thickening:** Add necessary edges.
  - Traverse remaining candidate edges in the order of decreasing evaluation.
  - Test for conditional independence in order to determine whether an edge is needed in the graphical model.
  - Use local Markov property to select a condition set: an attribute is conditionally independent of any non-neighbor given its neighbors.
  - Since the graphical model to be learned is undirected, *no (iterative) reduction of the condition set is needed* (decisive difference to Cheng–Bell–Liu Algorithm).

Learning Undirected Graphical Models Directly

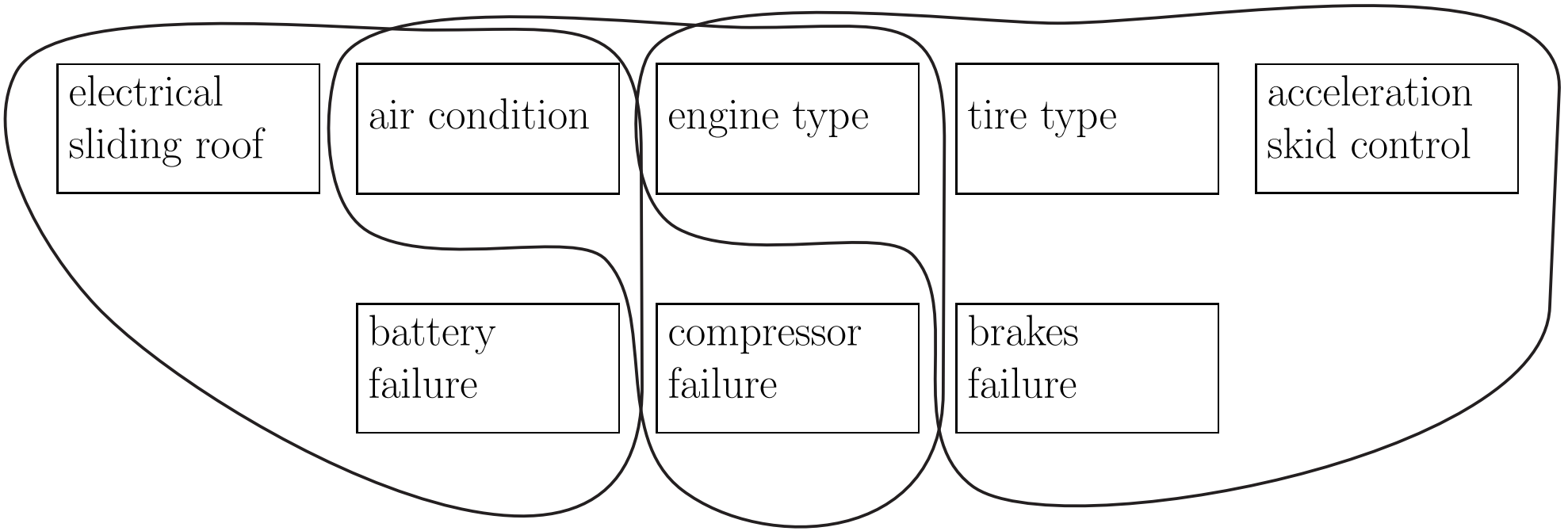
- **Moralizing:** Take care of possible v-structures.
  - If one assumes a perfect undirected map, this step is unnecessary. However, v-structures are too common and cannot be represented without loss in an undirected graphical model.
  - Possible v-structures can be taken care of by connecting the parents.
  - Traverse all edges with an evaluation below the independence threshold that have a common neighbor in the graph.
  - Add edge if conditional independence given the neighbors does not hold.
- **Thinning:** Remove superfluous edges.
  - In the thickening phase a conditional independence test may have failed, because the graph was still too sparse.
  - Traverse all edges that have been added to the current graphical model and test for conditional independence.

Application at Daimler AG

- **Improving the Product Quality by Detecting Weaknesses**
  - Learn a decision tree or inference network for vehicle properties and failures.
  - Look for suspicious conditional failure rates.
  - Find causes of these suspicious rates.
  - Optimize design of vehicle.
- **Improve the Error Diagnosis in Service Garages**
  - Learn a decision tree or inference network for vehicle properties and failures.
  - Record new faults.
  - Test for most probable errors.

Analysis of the Daimler Database

- Database: approx. 18500 vehicles with more than 100 attributes
- Analysis of dependencies between **specific equipment** and **failure**.
- Results are used as a starting point for technical investigation.



Fictitious example: There are significantly more battery failures, if an aircondition and an electrical sliding roof are installed.

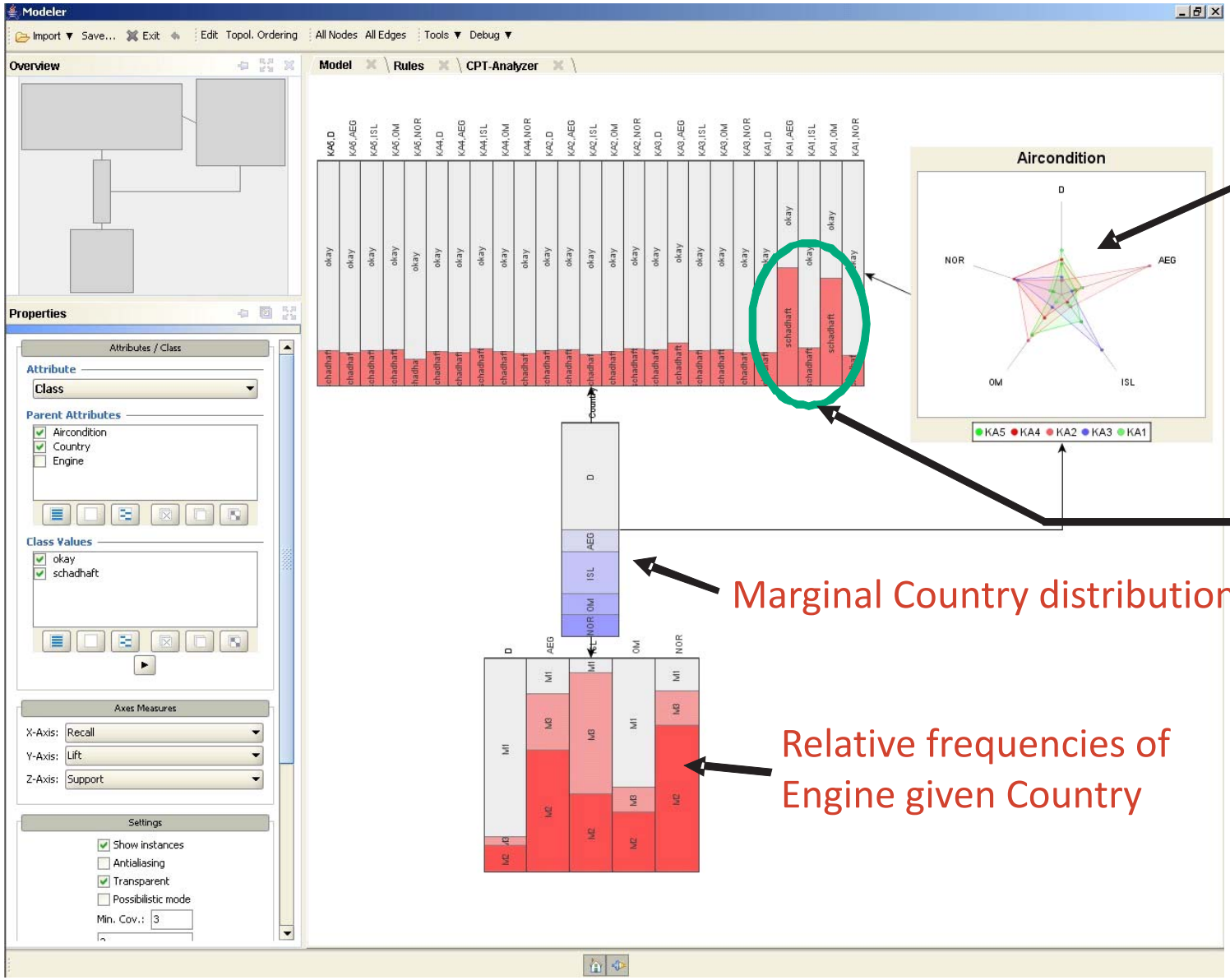
Example Network

Influence of specific equipment on battery failure:

(fictitious) battery failure rate		Aircondition	
		with	without
elec. sliding roof	with	8%	3%
	without	3%	2%

- Significant deviation from independent distribution.
- Hint for possible causes.
- Here: Larger battery might be required if both aircondition *and* electrical sliding roof are installed.

Explorative Data Analysis



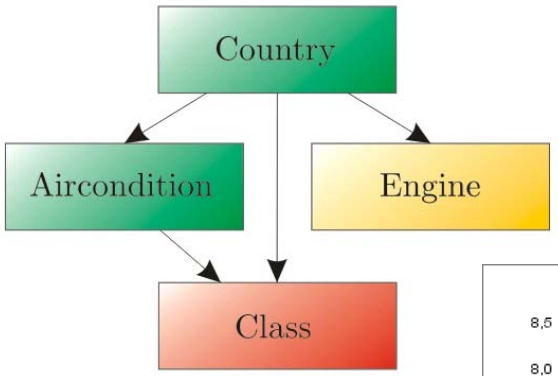
Ratio between (marginal) Aircondition sale rate and sale rate given the Country

Airconditions of type 1 fail much more often in Egypt and Oman

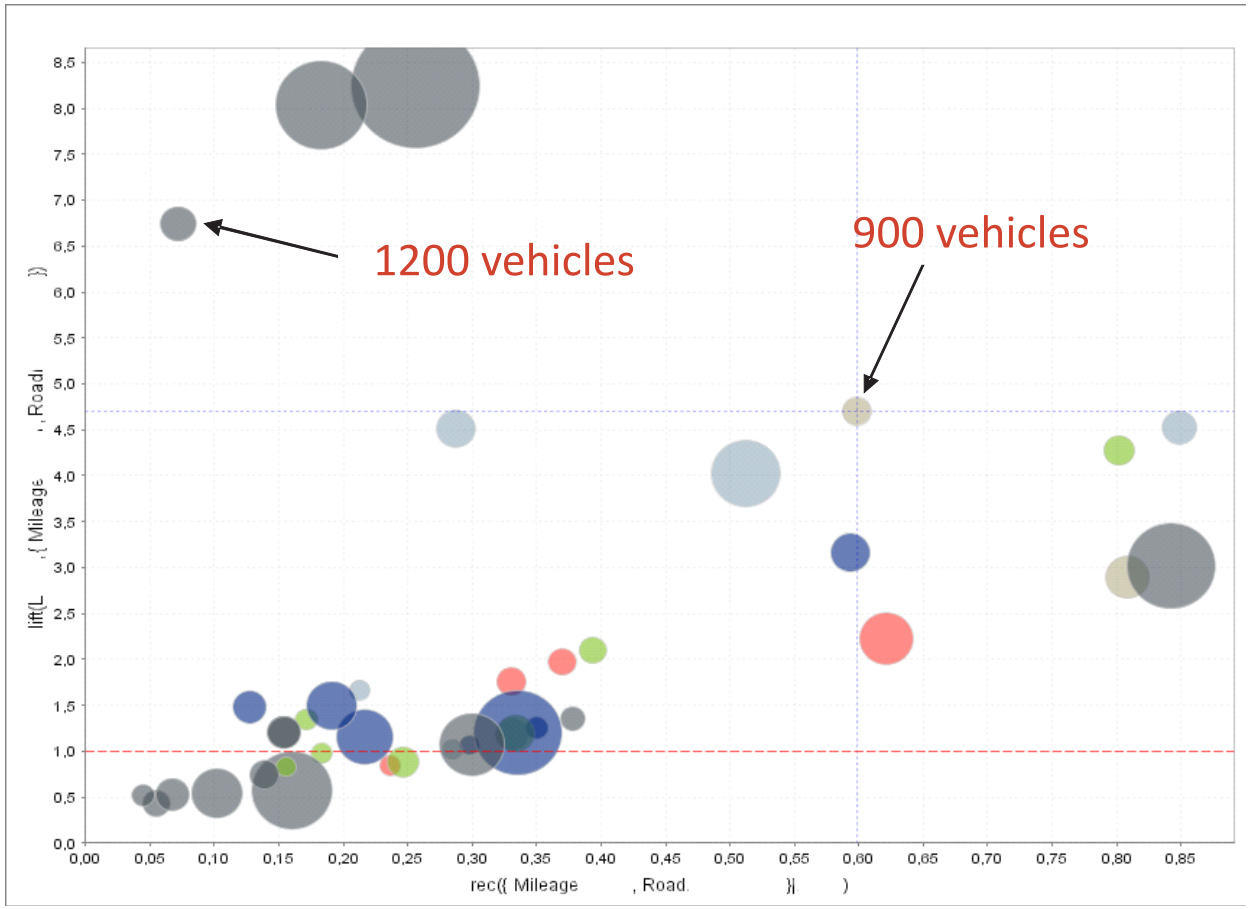
Marginal Country distribution

Relative frequencies of Engine given Country

Discovery of Local Patterns



Subnet of the entire network



Partition of the entire vehicle set according to the network's potential tables.