Algorithmisches Lernen/Machine Learning

Part 1: Stefan Wermter

- Introduction
- Connectionist Learning (e.g. Neural Networks)
- Decision-Trees, Genetic Algorithms

Part 2: Norman Hendrich

- Support-Vector Machines
- Learning of Symbolic Structures
- **Bayesian Learning (2)**
- Dimensionality Reduction

Part 3: Jianwei Zhang

- Function approximation
- Reinforcement Learning
- Applications in Robotics

Bayesian Learning

- Bayesian Reasoning
- Bayes Optimal Classifier
- Naïve Bayes Classifier
- Cost-Sensitive Decisions
- Modelling with Probability Density Functions
- Parameter Estimation
- Bayesian Networks
- Markov Models
- Dynamic Bayesian Networks

Bayesian/Belief Networks

- modelling all dependencies for a joint probability is impossible

  $$P(\mathcal{U}) = P(X_1|X_2, ..., X_n)P(X_2|X_3, ..., X_n)P(X_3|X_4, ..., X_n)P(X_n)$$

  - exponential in the number of variables

- reason: no independence assumptions
- ignoring the dependencies (naïve Bayes) is too strong a simplification

- goal: controlling the dependence/independence of variables

- recommended reading:
  Jensen, Finn V. and Nielsen, Thomas D. (2007) Bayesian Networks and Decision Graphs. Springer 2007.

Bayesian/Belief Networks

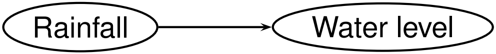
- causal reasoning: $P(X|Y_1, ..., Y_n)$
- causal reasoning works in both directions:

$$P(Waterlevel|Rainfall)$$

  - knowing that there was heavy/no rainfall will increase the belief that there will be a high/low water level
  - knowing there is a high/low water level will increase the belief that there was heavy/no rainfall

Bayesian/Belief Networks

- graphical models: variables are connected by edges if there is a causal relationship between them



- independence assumption for Bayesian networks:
  - a variable is independent of its non-descendants given its immediate predecessors

Bayesian/Belief Networks

- conditional independence:

  X is independent of Y given Z if

  $$\forall x_i \forall y_j \forall z_k . P(X = x_i | Y = y_j, Z = z_k) = P(X = x_i | Z = z_k)$$

- short:

  $$P(X|Y, Z) = P(X|Z)$$

- extention to sets of variables

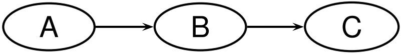
  $$P(X_1 ... X_l | Y_1 ... Y_m, Z_1 ... Z_n) = P(X_1 ... X_l | Z_1 ... Z_n)$$

Bayesian/Belief Networks

- three cases
  - sequences of causal influence
  - diverging connections
  - converging connections

Bayesian/Belief Networks

- sequences:



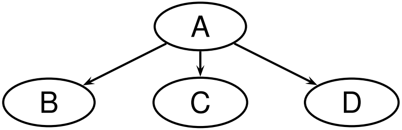
  - if B is instantiated the value of C is independent of the value of A
  - instantiating B blocks communication between A and C
  - A and C are d-separated given B

Bayesian/Belief Networks

- knowing that there was heavy rainfall will increase the belief that there will be a high water level and subsequently that there will be a flooding
- knowing there is a flooding will increase the belief that there is high water level and subsequently that there was heavy rainfall
- knowing that there is a high water level the additional knowledge about a flooding does not change the belief in heavy rainfall
- knowing that there is a high water level the additional knowledge about heavy rainfall does not change the belief in a flooding

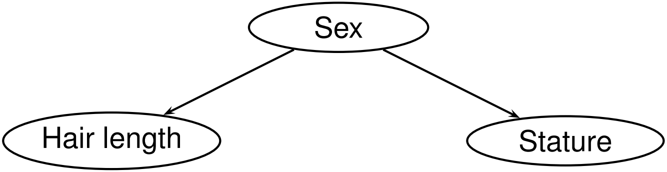
Bayesian/Belief Networks

- diverging connections:



- instantiating A blocks communication between B, C and D
- B, C, and D are d-separated given A

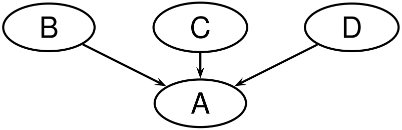
Bayesian/Belief Networks



- hair length gives evidence about the sex and the stature
- stature gives evidence about the sex and the hair length
- knowing the sex the additional knowledge about the hair length/the stature gives no additional knowledge about the stature/hair length

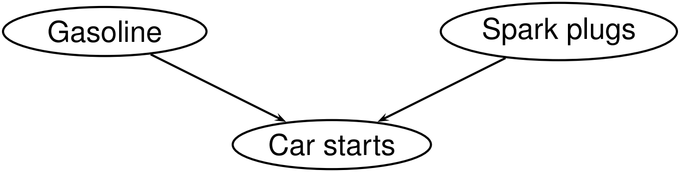
Bayesian/Belief Networks

- converging connections



- B, C, and D are not d-separated if either A or one of its descendants is instantiated
- no information flow if A is not instantiated
- ”explaining away” effect

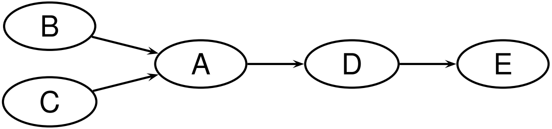
Bayesian/Belief Networks



- if no information on whether the car starts is available the information that the fuel tank is empty does not say anything about the state of the spark plugs
- if we know the car does not start the additional knowledge that the tank is empty/the spark plugs are dirty will decrease the belief that the spark plugs are dirty/the car is empty

Bayesian/Belief Networks

- evidence about a variable: statement about the certainty of its state (value)
- hard evidence: knowing the value (the variable is instantiated)
- soft evidence: otherwise



- hard evidence about E gives soft evidence about A
- soft evidence is sufficient for explaining away a reason
- blocking requires always hard evidence

Bayesian/Belief Networks

- a Bayesian/belief network is a joint probability distribution over a set of variables consisting
  - of a set of local conditional probabilities between the variables
  - together with a set of conditional independence assumptions

Bayesian/Belief Networks

- belief networks are represented by a directed acyclic graph

- nodes: variables with a finite set of mutually exclusive states (values)
- edges between nodes: modelling causal relationships
- conditional probability distributions for the values of each node A given the values of the parent nodes $B_1, ...B_n$ $P(A|B_1, ..., B_n)$
- if a node has no parents the conditional probabilities reduce to unconditional ones $P(A)$

Bayesian/Belief Networks

- if $P(\mathcal{U})$ is known every probability $P(A_i)$ or $P(A_i|e)$ can be computed.
    - $\mathcal{U} = \{A_1, A_2, ..., A_n\}$ is the universe of variables of a Bayesian network
    - e is evidence about some of the variables in the Bayesian network
- computing $P(\mathcal{U})$ is infeasible in the general case
    - great number of conditional probabilities which are impossible to estimate
    - naive computation of $P(\mathcal{U})$ is exponential in the number of variables

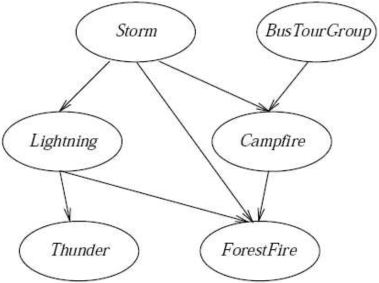
Bayesian/Belief Networks

- exploiting the independence assumptions captured by the network structure
- chain rule for Bayesian networks

$$P(\mathcal{U}) = \prod_{i=1}^{n} P(A_i|pa(A_i))$$

- A_i are variables
- $P(A_i|...)$ (conditional) probability distributions (potentials)

Modelling with Bayesian Networks

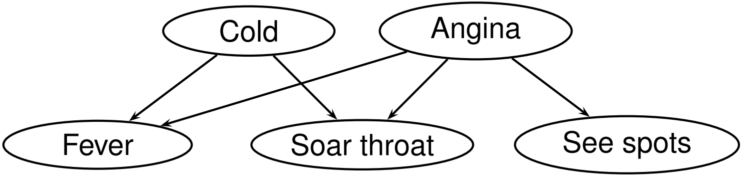


	S,B	$S,\neg B$	$\neg S,B$	$\neg S,\neg B$
C	0.4	0.1	0.8	0.2
$\neg C$	0.6	0.9	0.2	0.8

- causal reasoning: the probability of *Campfire* depends on *Storm*, and *BusTourGroup* , *and nothing else*

Modelling with Bayesian Networks

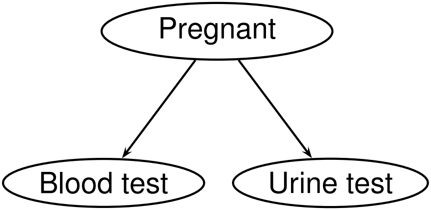
- direct influence



- causal model: generates the observations

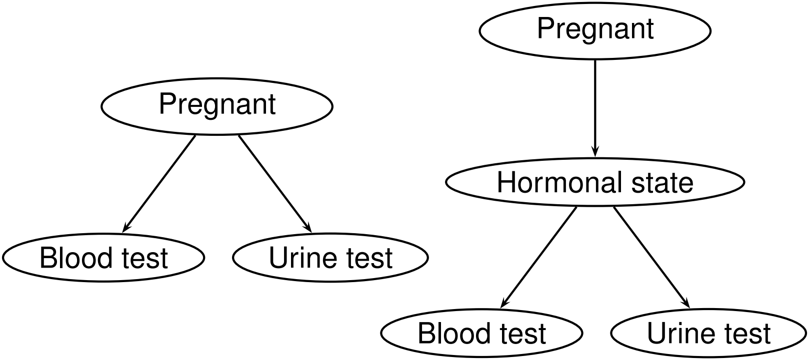
Modelling with Bayesian Networks

- indirect influence



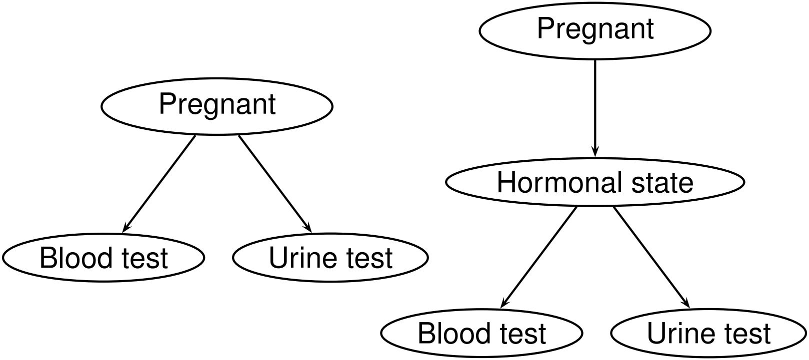
Modelling with Bayesian Networks

- indirect influence



Modelling with Bayesian Networks

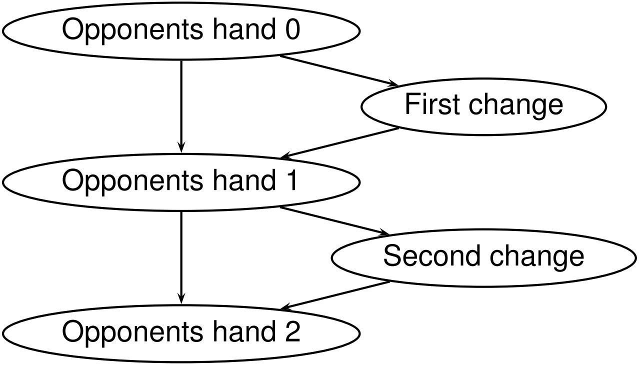
- indirect influence



- mediating variables

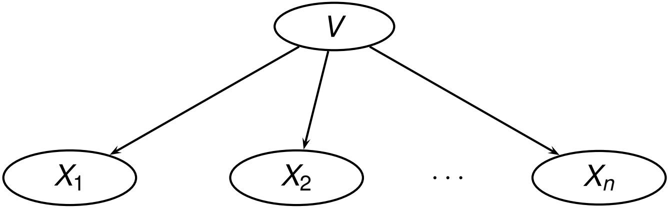
Modelling with Bayesian Networks

- temporal sequences: Poker game



Modelling with Bayesian Networks

- Naïve Bayes model



Inference in Bayesian Networks

- general case: computing the probability distribution of any subset of variables given the values or distributions for any subset of the remaining variables
- special case: computing the probability distribution of a variable given the values or distributions for the remaining variables

Inference in Bayesian Networks

- computation of a probability means marginalizing out all the other variables from the joint probability of a variable assignment ...

$$P(A_i) = \sum_{A_j, j \neq i} \prod_{j=1}^{n} P(A_j | parents(A_j))$$

- ... without computing the joint probability distribution

$$P(A_1, ..., A_n) = \prod_{i=1}^{n} P(A_i | parents(A_i))$$

- tractability requirement: keep the conditional probability distributions of intermediate results as small as possible

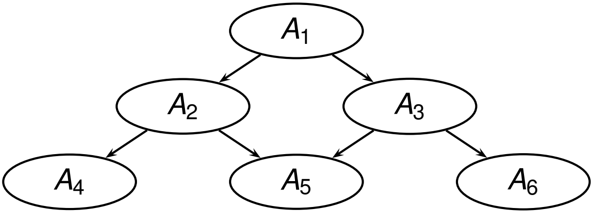
Inference in Bayesian Networks

- marginalizing out a variable

$$P(A_1) = \sum_{A_2} P(A_1, A_2)$$

$$P(A_i) = \sum_{A_j, j \neq i} P(A_1, ..., A_n)$$

Inference in Bayesian Networks



$$P(A_4) = \sum_{A_1,A_2,A_3,A_5,A_6} P(\mathcal{U})$$

$$= \sum_{A_1,A_2,A_3,A_5,A_6} \prod_{j=1}^{n} P(A_j|parents(A_j))$$

$$= \sum_{A_1,A_2,A_3,A_5,A_6} P(A_1)P(A_2|A_1)P(A_3|A_1)P(A_4|A_2)$$

$$P(A_5|A_2,A_3)P(A_6|A_3)$$

Inference in Bayesian Networks

- distributive law for probability distributions

$$\sum_A P(B|...)P(C|...) = P(B|...) \sum_A P(C|...) \qquad A \notin dom(P(B|...))$$

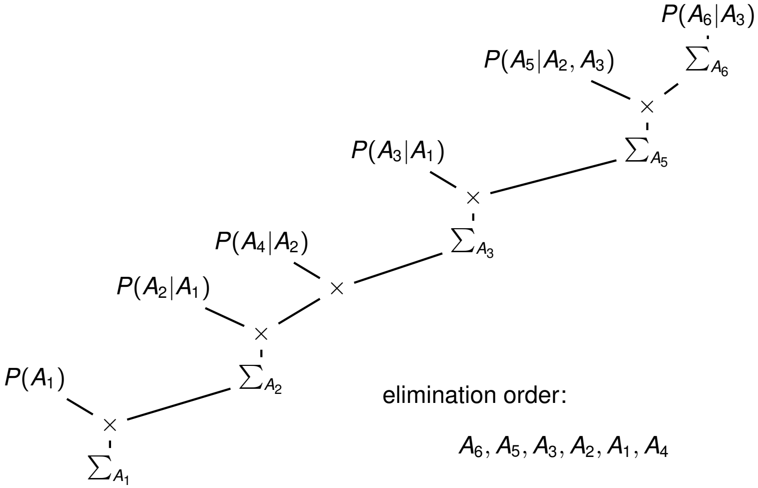
$$dom(P(A|B_1, ..., B_n)) = \{A, B_1, ..., B_n\}$$

Inference in Bayesian Networks

$$
\begin{aligned}
P(A_4) &= \sum_{A_1,A_2,A_3,A_5,A_6} \prod_{j=1}^{n} P(A_j|parents(A_j)) \\
&= \sum_{A_1,A_2,A_3,A_5,A_6} P(A_1)P(A_2|A_1)P(A_3|A_1)P(A_4|A_2)P(A_5|A_2,A_3)P(A_6|A_3) \\
&= \sum_{A_1} P(A_1) \sum_{A_2,A_3,A_5,A_6} P(A_2|A_1)P(A_3|A_1)P(A_4|A_2)P(A_5|A_2,A_3)P(A_6|A_3) \\
&= \sum_{A_1} P(A_1) \sum_{A_2} P(A_2|A_1) \sum_{A_3,A_5,A_6} P(A_3|A_1)P(A_4|A_2)P(A_5|A_2,A_3)P(A_6|A_3) \\
&= \sum_{A_1} P(A_1) \sum_{A_2} P(A_2|A_1) \sum_{A_3} P(A_3|A_1) \sum_{A_5,A_6} P(A_4|A_2)P(A_5|A_2,A_3)P(A_6|A_3) \\
&= \sum_{A_1} P(A_1) \sum_{A_2} P(A_2|A_1) \sum_{A_3} P(A_3|A_1)P(A_4|A_2) \sum_{A_5,A_6} P(A_5|A_2,A_3)P(A_6|A_3) \\
&= \sum_{A_1} P(A_1) \sum_{A_2} P(A_2|A_1)P(A_4|A_2) \sum_{A_3} P(A_3|A_1) \sum_{A_5,A_6} P(A_5|A_2,A_3)P(A_6|A_3) \\
&= \sum_{A_1} P(A_1) \sum_{A_2} P(A_2|A_1)P(A_4|A_2) \sum_{A_3} P(A_3|A_1) \sum_{A_5} P(A_5|A_2,A_3) \sum_{A_6} P(A_6|A_3)
\end{aligned}
$$

Inference in Bayesian Networks

$$P(A_4) = \sum_{A_1} P(A_1) \sum_{A_2} P(A_2|A_1) P(A_4|A_2) \sum_{A_3} P(A_3|A_1) \sum_{A_5} P(A_5|A_2, A_3) \sum_{A_6} P(A_6|A_3)$$



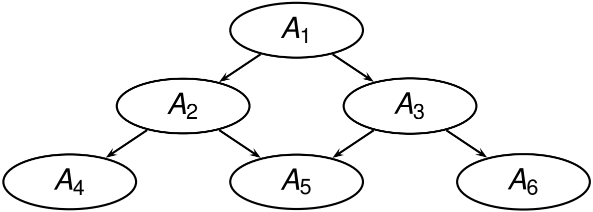
elimination order:

$A_6, A_5, A_3, A_2, A_1, A_4$

Inference in Bayesian Networks

- usually several alternative elimination orders
- goal: determining the optimal elimination order (for all variables)

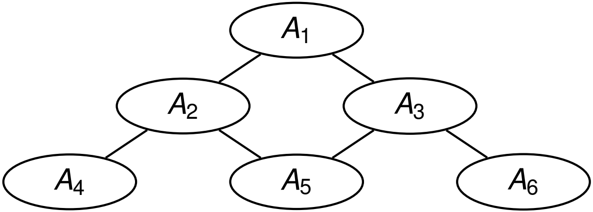
- domain graph: connects all variables which appear together in a domain of a probability distribution
- contains all edges of the Bayesian network plus connections between nodes which share a common child node



Inference in Bayesian Networks

- usually several alternative elimination orders
- goal: determining the optimal elimination order (for all variables)

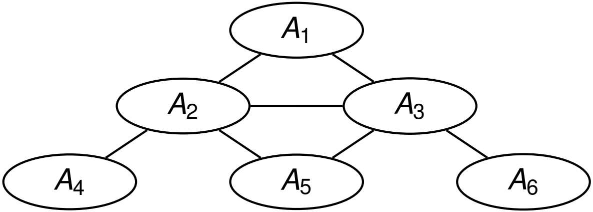
- domain graph: connects all variables which appear together in a domain of a probability distribution
- contains all edges of the Bayesian network plus connections between nodes which share a common child node



Inference in Bayesian Networks

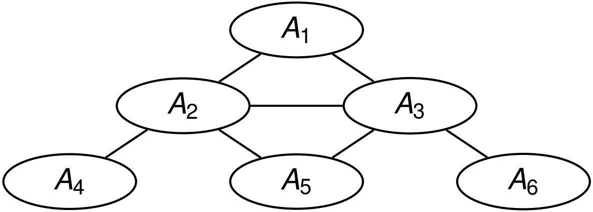
- usually several alternative elimination orders
- goal: determining the optimal elimination order (for all variables)

- domain graph: connects all variables which appear together in a domain of a probability distribution
- contains all edges of the Bayesian network plus connections between nodes which share a common child node



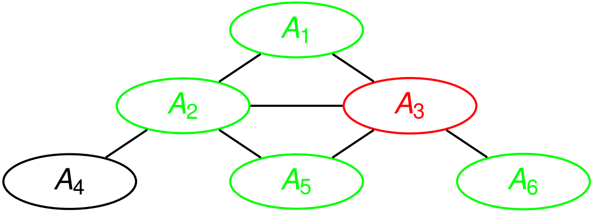
Inference in Bayesian Networks

- perfect elimination sequence: elimination sequence which does not produce additional links (fill-ins) in the domain graph
  - it avoids computing new distributions



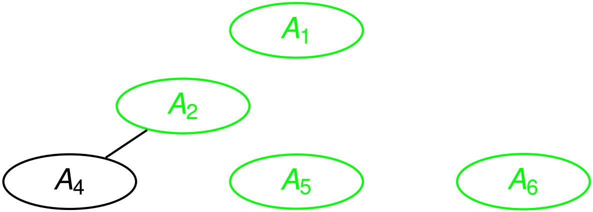
Inference in Bayesian Networks

- perfect elimination sequence: elimination sequence which does not produce additional links (fill-ins) in the domain graph
  - it avoids computing new distributions



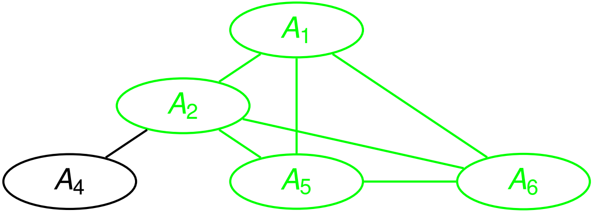
Inference in Bayesian Networks

- perfect elimination sequence: elimination sequence which does not produce additional links (fill-ins) in the domain graph
  - it avoids computing new distributions



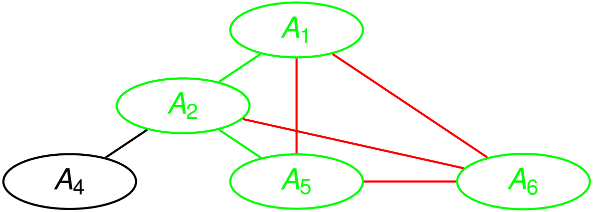
Inference in Bayesian Networks

- perfect elimination sequence: elimination sequence which does not produce additional links (fill-ins) in the domain graph
  - it avoids computing new distributions



Inference in Bayesian Networks

- perfect elimination sequence: elimination sequence which does not produce additional links (fill-ins) in the domain graph
  - it avoids computing new distributions



Inference in Bayesian Networks

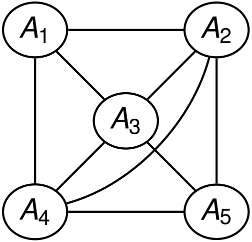
- perfect elimination sequences for the example

$$
\begin{aligned}
&A_6, A_5, A_3, A_1, A_2, A_4 \\
&A_5, A_6, A_3, A_1, A_2, A_4 \\
&A_1, A_5, A_6, A_3, A_2, A_4 \\
&A_6, A_1, A_3, A_5, A_2, A_4
\end{aligned}
$$

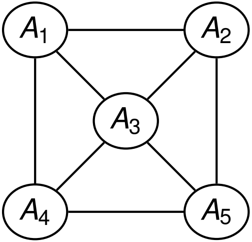
- a perfect elimination sequence ending in variable A is optimal with respect to calculating $P(A)$
- complete task: find an optimal elimination sequence for each variable in $\mathcal{U}$

Inference in Bayesian Networks

- triangulated graph: graph which contains a perfect elimination sequence



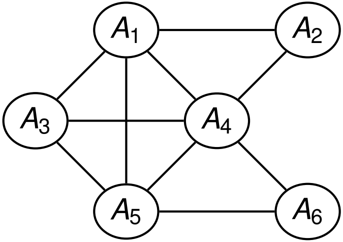
triangulated

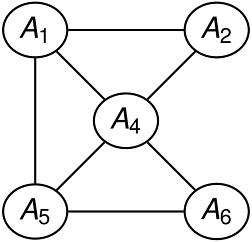


not triangulated

Inference in Bayesian Networks

- checking for a perfect elimination sequence: successively eliminate simplical nodes from the graph until all nodes have been removed
- simplical node: node with a complete neighbor set
- complete set: all nodes are pairwise connected



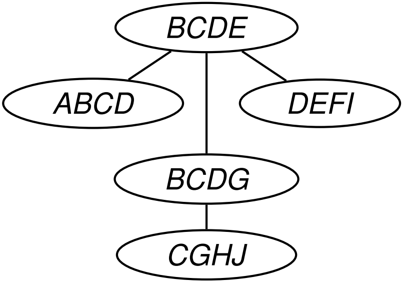


Inference in Bayesian Networks

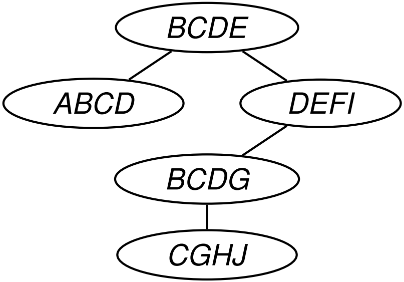
- clique: complete set which is not a subset of another complete set, i.e. a maximal complete set
- a node X is simplical iff its familiy $fa(X)$ is a clique
- an undirected graph is triangulated iff all nodes can be eliminated by successively eliminate simplical nodes
- procedure for finding a clique
  1. eliminate a simplical node A if $fa(A)$ is a clique candidate
  2. if $fa(A)$ does not contain all the remaining nodes continue with 1
  3. prune the set of clique candidates by removing all sets that are subsets of other clique candidates

Inference in Bayesian Networks

- join tree:
  - nodes: cliques of a graph
  - all nodes on a path between two nodes V and W contain the intersection $V \cap W$
- if the cliques of a graph can be arranged as a join tree, the graph is triangulated



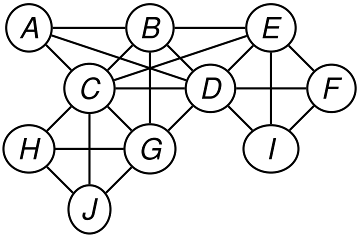
a join tree

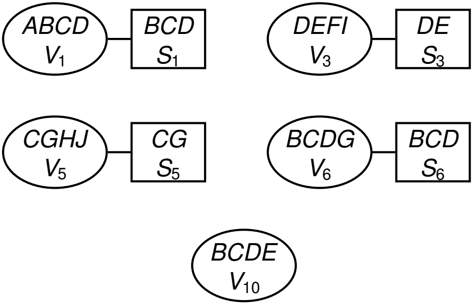


not a join tree

Inference in Bayesian Networks

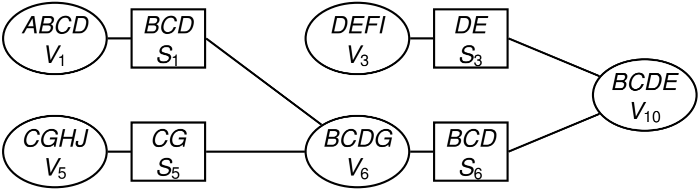
- procedure for constructing a join tree
  1. start with a simplical node X, i.e. $fa(X)$ is a clique
  2. remove nodes from $fa(X)$ that have neighbors only in $fa(X)$
  3. $fa(X)$ receives an index according to the number of nodes removed so far
  4. the set of remaining nodes of $fa(X)$ is called a separator
  5. continue with 1 until all the cliques have been removed





Inference in Bayesian Networks

- procedure for constructing a join tree (2)
  - connect each separator S_i to a clique V_j such that $j > i$ and $S_i \subset V_j$



Inference in Bayesian Networks

- triangulation of graphs
  - eliminate simplical nodes
  - if the remaining graph does not contain a simplical node choose an arbitrary node and make its family complete by adding fill-ins
- non-deterministic choice
- heuristics: eliminate the node with minimal

$$sz(fa(X)) = \prod_{Y \in fa(X)} |sp(Y)|$$

$sp(X)$: number of values (states) of variable X

Inference in Bayesian Networks

- updating probabilities after receiving evidence

$$P(\mathcal{U}, e) = \prod_{A \in \mathcal{U}} P(A|pa(A)) \prod_{i=1}^{m} e_i$$

- e_i is a vector over $\{0, 1\}$ associated with a particular node, specifying which states (values) are possible/impossible

$$P(A|e) = \frac{\sum_{\mathcal{U} \setminus \{A\}} P(\mathcal{U}, e)}{P(e)}$$

Learning of Bayesian Networks

- estimating the probabilities for a given structure
  - for complete data:
    - maximum likelihood estimation
    - Bayesian estimation
  - for incomplete data
    - expectation maximization
    - gradient descent methods
- learning the network structure

Learning of Bayesian Networks

- expectation maximization
  - calculate the table of expected counts

$$\underset{\Theta^t}{\boldsymbol{E}}\left(N(X_i, pa(X_i))|\mathcal{D}\right) = \sum_{d \in \mathcal{D}} P(X_i, pa(X_i)|d, \Theta^t)$$

  - use the expected counts as if they were actual counts to compute a new likelihood estimate for Θ

$$\hat{\Theta}_{ijk} = \frac{\boldsymbol{E}_{\Theta^t}(N(X_i = k), pa(X_i) = j)|\mathcal{D}}{\sum_{h=1}^{|sp(X_i)|} \boldsymbol{E}_{\Theta^t}(N(X_i = h, pa(X_i) = j|\mathcal{D})}$$

$|sp(X)|$: number of values of X

Learning of Bayesian Networks

- learning the network structure
- space of possible networks is extremely large ($> \mathcal{O}(2^n)$)
- a Bayesian network over a complete graph is always a possible answer, but not an interesting one (no modelling of independencies)
- problem of overfitting
- two apporaches
  - constraint-based learning
  - (score-based learning)

Learning of Bayesian Networks

- constraint-based structure learning
  - estimate the pairwise degree of independence using conditional mutual information
  - determine the direction of the arcs between non-independent nodes

Learning of Bayesian Networks

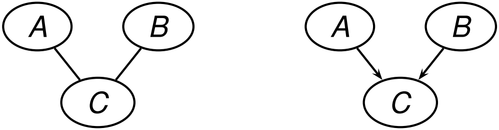
- conditional mutual information

$$CMI(A, B|\mathcal{X}) = \sum_{\mathcal{X}} \widehat{P}(\mathcal{X}) \sum_{A,B} \widehat{P}(A, B|\mathcal{X}) log_2 \frac{\widehat{P}(A, B|\mathcal{X})}{\widehat{P}(A|\mathcal{X})\widehat{P}(B|\mathcal{X})}$$

- two nodes are independent if $CMI(A, B|\mathcal{X}) = 0$
- choose all pairs of nodes as non-independent, where the significance of a χ^2-test on the hypothesis $CMI(A, B|\mathcal{X}) = 0$ is above a certain user-defined threshold
- high minimal significance level: more links are established
- result is a skeleton of possible candidates for causal influence

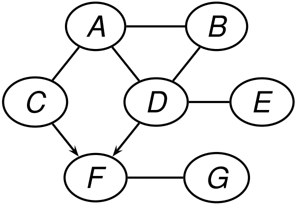
Learning of Bayesian Networks

- determining the direction of the causal influence
  - Rule 1 (introduction of v-structures): $A - C$ and $B - C$ but not $A - B$ introduce a v-structure $A \rightarrow C \leftarrow B$ if there exists a set of nodes $\mathcal{X}$ so that A is d-separated from B given $\mathcal{X}$

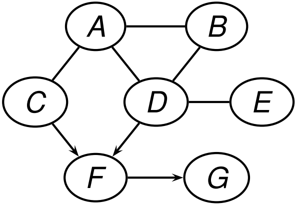


  - Rule 2 (avoid new v-structures): When Rule 1 has been exhausted and there is a structure $AtoC - B$ but not $A - B$ then direct $C \rightarrow B$
  - Rule 3 (avoid cycles): If $A \rightarrow B$ introduces a cycle in the graph do $A \leftarrow B$
  - Rule 4 (choose randomly): If no other rule can be applied to the graph, choose an undirected link and give it an arbitrary direction

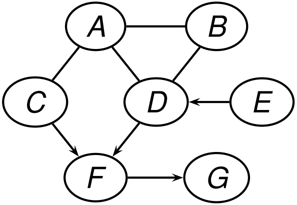
Learning of Bayesian Networks



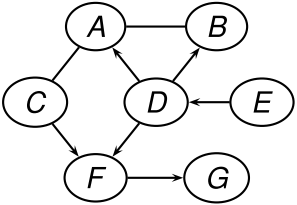
Rule 1



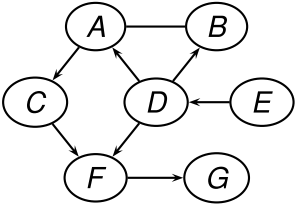
Rule 2



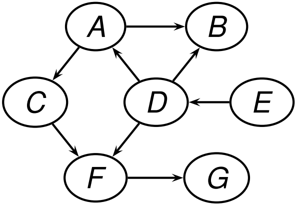
Rule 4



Rule 2



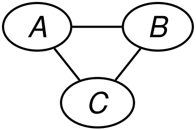
Rule 2

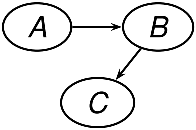


Rule 4

Learning of Bayesian Networks

- independence/non-independence candidates might contradict each other
- $\neg I(A,B), \neg I(A,C), \neg I(B,C)$, but $I(A,B|C), I(A,C|B)$ and $I(B,C|A)$
  - remove a link and build a chain out of the remaining ones

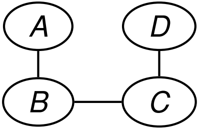


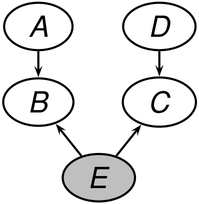


  - uncertain region: different heuristics might lead to different structures

Learning of Bayesian Networks

- $I(A, C), I(A, D), I(B, D)$





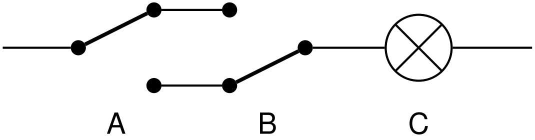
  - problem might be caused by a hidden variable $E \rightarrow B$ $E \rightarrow C$ $A \rightarrow B$ $D \rightarrow C$

Learning of Bayesian Networks

- useful results can only be expected, if
  - the data is complete
  - no (unrecognized) hidden variables obscure the induced influence links
  - the observations are a faithful sample of an underlying Bayesian network
    - the distribution of cases in $\mathcal{D}$ reflects the distribution determined by the underlying network
    - the estimated probability distribution is very close to the underlying one
  - the underlying distribution is recoverable from the observations

Learning of Bayesian Networks

- example of an unrecoverable distribution:
  - two switches: $P(A = up) = P(B = up) = 0.5$
  - $P(C = on) = 1$ if $val(A) = val(B)$
  - $\rightarrow I(A, C), I(B, C)$



- problem: independence decisions are taken on individual links (CMI), not on complete link configurations

$$P(C|A, B) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Bayesian Learning

- Bayesian Reasoning
- Bayes Optimal Classifier
- Naïve Bayes Classifier
- Cost-Sensitive Decisions
- Modelling with Probability Density Functions
- Parameter Estimation
- Bayesian Networks
- Markov Models
- Dynamic Bayesian Networks

Markov Models

- Markov Models (n-gram)
- Hidden Markov Models
- Training of Hidden Markov Models

Markov Models

- Markov Models (n-gram)
- Hidden Markov Models
- Training of Hidden Markov Models

Markov Models

- special case of Bayesian/belief networks for describing sequential observations
- modelling dependencies of various lengths
  - bigrams: $P(y_i|y_{i-1})$
  - trigrams: $P(y_i|y_{i-2}y_{i-1})$
  - quadrograms: $P(y_i|y_{i-3}y_{i-2}y_{i-1})$
  - ...
- e.g. to predict the probability of the next event
- speech and language processing, genome analysis, time series predictions (stock market, natural desasters, ...)

Markov Models

- examples of Markov chains for German letter sequences
- unigrams

  aiobnin*tarsfneonlpiitdregedcoa*ds*e*dbieastnreleeucdkeaitb* dnurlarsls*omn*keu**svdleeoieei* ...
- bigrams

  er*agepteprteiningeit*gerelen*re*unk*ves*mterone*hin*d*an* nzerurbom* ...
- trigrams

  billunten*zugen*die*hin*se*sch*wel*war*gen*man*nicheleblant* diertunderstim* ...
- quadrograms

  eist*des*nich*in*den*plassen*kann*tragen*was*wiese*zufahr* ...

Markov Models

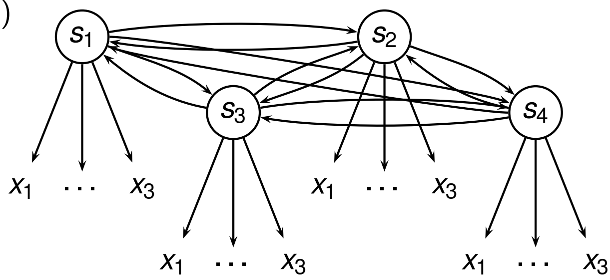
- Markov Models (n-gram)
- Hidden Markov Models
- Training of Hidden Markov Models

Hidden Markov Models

- symbol strings are usually fully observable
  $\rightarrow$ estimating the probabilities by counting and normalizing
- observation may depend on a underlying, not observable stochastic process
  $\rightarrow$ Hidden Markov Models

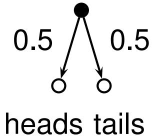
Hidden Markov Models

- Hidden Markov Model: doubly stochastic process
  - state transitions $P_t(s_i|s_{i-1})$: states change randomly
  - emission of symbols from states $p_e(\vec{x}|s_i)$: observations are generated randomly
  - initial state: $P_i(s_i)$



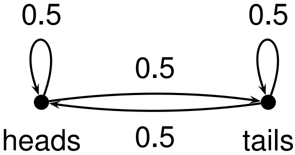
Hidden Markov Models

- Hidden Markov Models are able to capture the same regularities with vastly different probability estimations
  $\rightarrow$ high flexibility to accomodate unknown regularities
- example: coin
- emission probability only



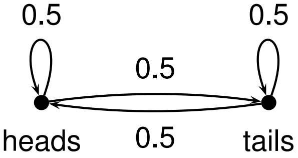
Hidden Markov Models

- transition probabilities only (1st order Markov model)

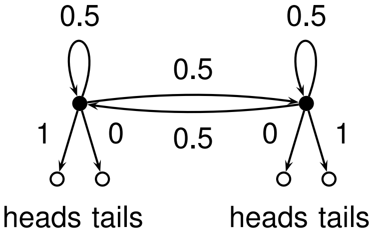


Hidden Markov Models

- transition probabilities only (1st order Markov model)

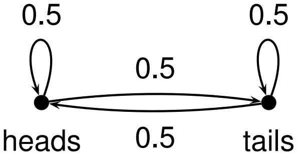


- Hidden Markov Models for the observation

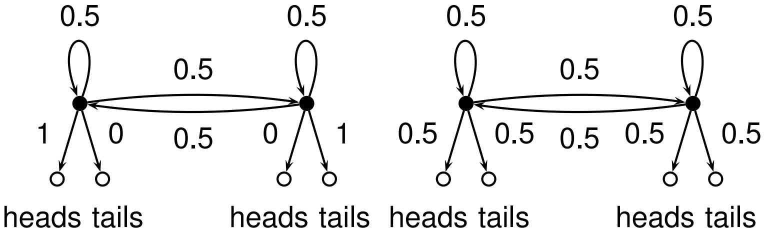


Hidden Markov Models

- transition probabilities only (1st order Markov model)

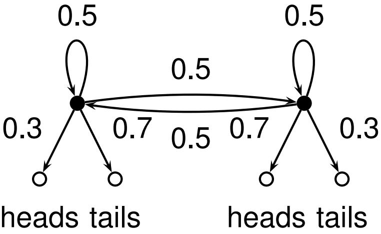


- Hidden Markov Models for the observation



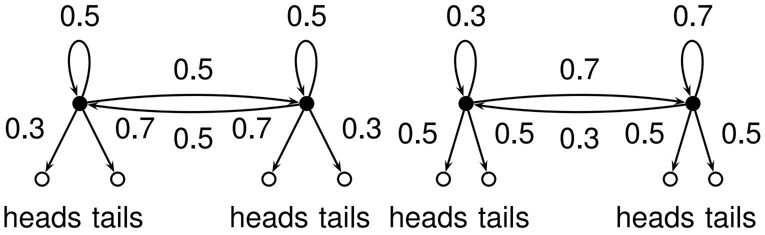
Hidden Markov Models

- alternative HMMs for the same observation



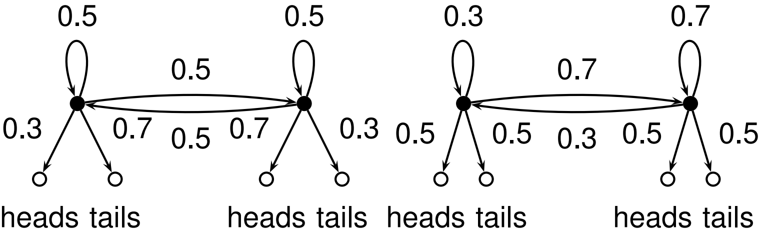
Hidden Markov Models

- alternative HMMs for the same observation



Hidden Markov Models

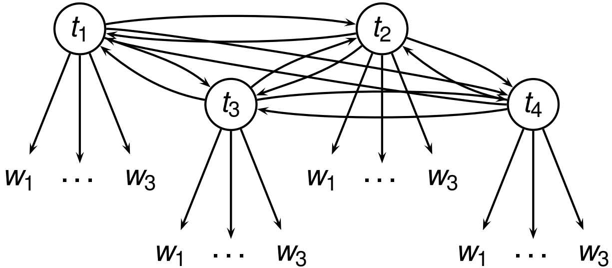
- alternative HMMs for the same observation



- even more possibilities for biased coins or coins with more than two sides

Hidden Markov Models

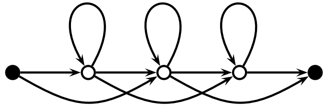
- example: part-of-speech tagging



- sequence labelling problem
- one-to-one correspondence between states and tags
- typical case: trigram transition probabilities
- emission of words depending on the state

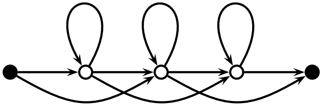
Hidden Markov Models

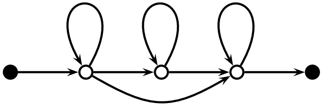
- example: speech recognition
- subsequences of observations are mapped to one label
- model topologies for phones (only transitions depicted)



Hidden Markov Models

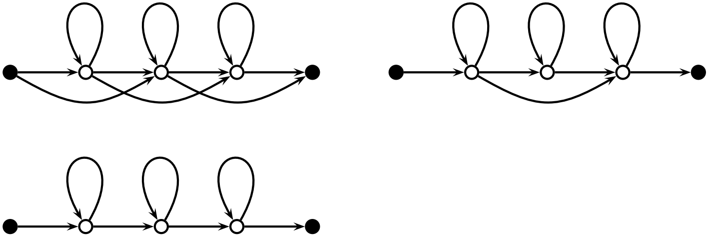
- example: speech recognition
- subsequences of observations are mapped to one label
- model topologies for phones (only transitions depicted)





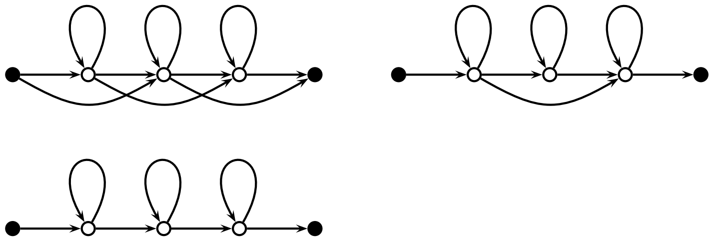
Hidden Markov Models

- example: speech recognition
- subsequences of observations are mapped to one label
- model topologies for phones (only transitions depicted)



Hidden Markov Models

- example: speech recognition
- subsequences of observations are mapped to one label
- model topologies for phones (only transitions depicted)



the more data available $\rightarrow$ the more sophisticated models can be trained

Markov Models

- Markov Models (n-gram)
- Hidden Markov Models
- Training of Hidden Markov Models

Training of Hidden Markov Models

- special case of EM: Baum-Welch training
  - start with an initial parameter set
  - iteratively improve the estimation
- converges to a local maximum
- no prior segmentation/alignment of the sequence required
- can be combined with the estimation of mixture densities

Training of Hidden Markov Models

- forward coefficients: $\alpha_n(i)$
- probability for producing a partial sequence $x[1{:}n]$ by a path leading to state s_i

$$\alpha_n(i) = p(x[1{:}n], l_n = s_i | \mathcal{M})$$

- initialization

$$\alpha_1(i) = P_i(s_i)\ p_e(x[1]|s_i)$$

- induction

$$\alpha_{n+1}(j) = p_e(x[n+1]|s_j)\ \sum_{i=1}^{I} \alpha_n(i)\ P_t(s_j|s_i)$$

- probability of the whole input sequence

$$p(x[1{:}N]|\mathcal{M}) = \sum_{i=1}^{I} \alpha_N(i)$$

Training of Hidden Markov Models

- backward coefficients: $\beta_n(i)$
- probability to leave a state on a certain path

$$\beta_n(i) = p(x[n{:}N] | s_i = l_n, \mathcal{M})$$

$$\beta_N(i) = 1$$

$$\beta_n(j) = \sum_{i=1}^{I} P_t(s_i|s_j)\, p_e(x[n+1]|s_i)\, \beta_{n+1}(i)$$

Training of Hidden Markov Models

- $\gamma_n(i)$: probability of the model $\mathcal{M}$ to be in state s_i at a certain point in time

$$\gamma_n(i) = p(l_n = s_i | x[1{:}N], \mathcal{M}) = \frac{\alpha_n(i)\ \beta_n(i)}{p(x[1{:}N]|\mathcal{M})}$$

- $\xi_n(i,j)$: probability of a transition from state s_i to state s_j given the training data

$$\begin{aligned} \xi_n(i,j) &= p(l_l = s_i, l_{l+1} = s_j | x[1{:}N], \mathcal{M}) \\ &= \frac{\alpha_n(i)\ P_t(s_j|s_i)\ p_e(x[n{+}1]|z_j)\ \beta_{n+1}(j)}{p(x[1{:}N]|\mathcal{M})} \end{aligned}$$

Training of Hidden Markov Models

- EM re-estimation

$$p_i'(s_i) = \gamma_1(i)$$

$$P_t'(s_j|s_i) = \frac{\sum_{n=1}^{N-1} \xi_n(i,j)}{\sum_{n=1}^{N-1} \gamma_n(i)}$$

$$p_e'(x|s_i) = \frac{\left[\sum_{n=1}^{N} \gamma_n(i)\right]_{x[n]=x}}{\sum_{n=1}^{N} \gamma_n(i)}$$

Bayesian Learning

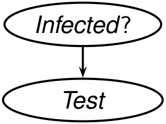
- Bayesian Reasoning
- Bayes Optimal Classifier
- Naïve Bayes Classifier
- Cost-Sensitive Decisions
- Modelling with Probability Density Functions
- Parameter Estimation
- Bayesian Networks
- Markov Models
- Dynamic Bayesian Networks

Dynamic Bayesian Networks

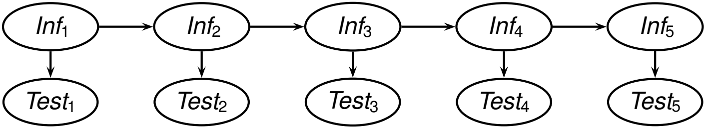
- modelling sequences of observations
- representing individual time slices and their connections to neighboring time slices
- enrolling the time slices according to the length of the observation sequence
  - initial segment
  - middle segment
  - final segment
- Markov property: links have a limited time horizon (e.g. from the previous slice to the current one)

Dynamic Bayesian Networks

- example: milk infection test
- the probability of the test outcome depends on the cow being infected or not

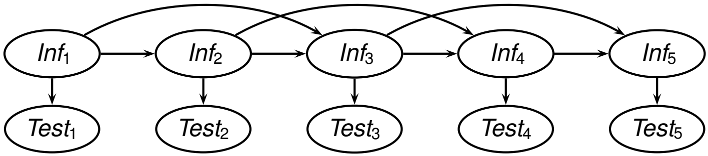


- the probability of the cow being infected depends on the cow being infected on the previous day
  - first order Markov model



Dynamic Bayesian Networks

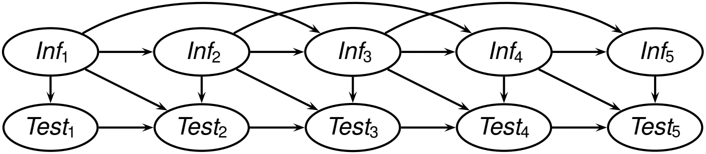
- the probability of the cow being infected depends on the cow being infected on the two previous days
  - incubation and infection periods of more than one day
  - second order Markov model



  - assumes only random test errors

Dynamic Bayesian Networks

- the probability of the test outcome also depends on the cow's health and the test outcome on the previous day
  - can also capture systematic test errors
  - second order Markov model for the infection
  - first order Markov model for the test results



Dynamic Bayesian Networks

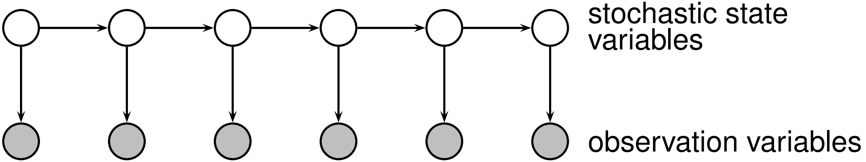
- relationship between HMM and DBN

	HMM	DBN
nodes	states	variables
edges	state transitions	causal influence
# nodes	# model states	length of the observation sequence

- causal links can be stochastic *or* deterministic
  - stochastic: conditional probabilities to be estimated
  - deterministic: to be specified manually (decision trees)

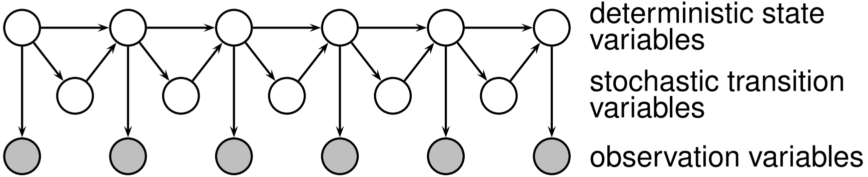
Dynamic Bayesian Networks

- modelling the state of the model: setting a state variable to a certain state number
- changing the state of the model: setting a state variable in slice i according to values in slice $i-1$



Dynamic Bayesian Networks

- alternative model structure: separation of state and transition variables



Dynamic Bayesian Networks

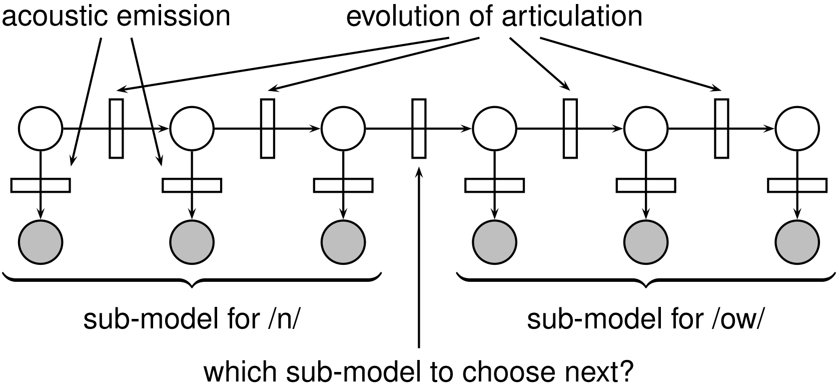
- state variables
  - distinct values for each state of the corresponding HMM
  - value at slice $t + 1$ is a deterministic function of the state and the transition of slice t
- transition variables
  - probability distribution
  - which arc to take to leave a state of the corresponding HMM
  - number of values is the outdegree of the corresponding state in an HMM
- use of transition variables is more efficient than using stochastic state variables with zero probabilities for the impossible state transitions

Dynamic Bayesian Networks

- composite models: some applications require the model to be composed out of sub-models
  - speech: phones $\rightarrow$ syllables $\rightarrow$ words $\rightarrow$ utterances
  - vision: sub-parts $\rightarrow$ parts $\rightarrow$ composites
  - genomics: nucleotides $\rightarrow$ amino acids $\rightarrow$ proteins

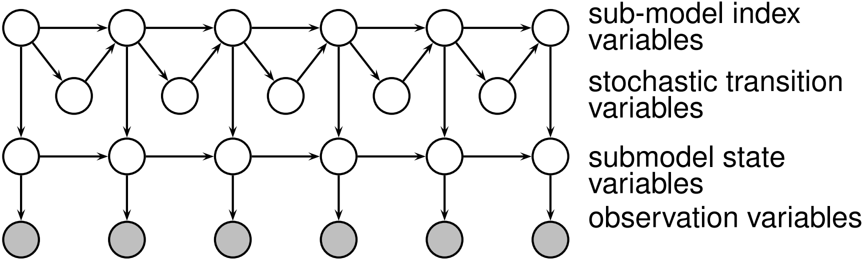
Dynamic Bayesian Networks

- composite models:
  - length of the sub-segments is not kown in advance
  - naive concatenation would require to generate all possible segmentations of the input sequence



Dynamic Bayesian Networks

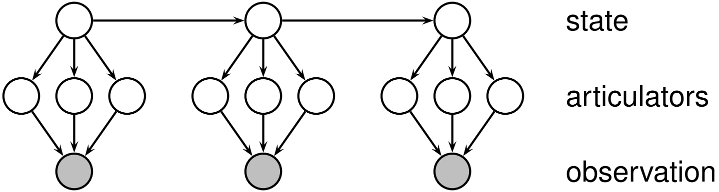
- additional sub-model variables select the next sub-model to choose



- sub-model index variables: which submodel to use at each point in time
- sub-model index and transition variables model legal sequences of sub-models (control layer)
- several control layers can be combined

Dynamic Bayesian Networks

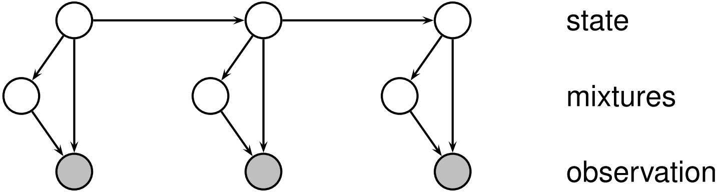
- factored models (1): factoring out different influences on the observation
- e.g. articulation:
  - asynchroneous movement of articulators (lips, tongue, jaw, ...)



- if the data is drawn from a factored source, DBNs are superior to HMMs

Dynamic Bayesian Networks

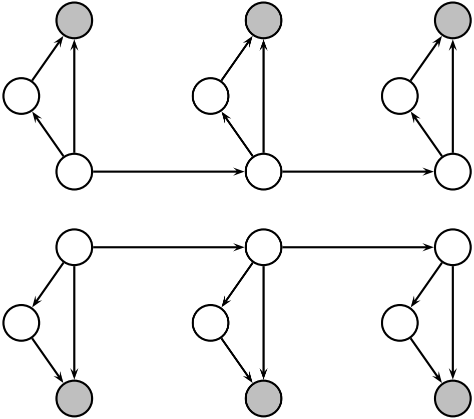
- factored models (2): coupling of different input channels
  - e.g. acoustic and visual information in speech processing
- naïve approach (1): data level fusion



- too strong synchronisation constraints

Dynamic Bayesian Networks

- naïve approach(2): independent input streams



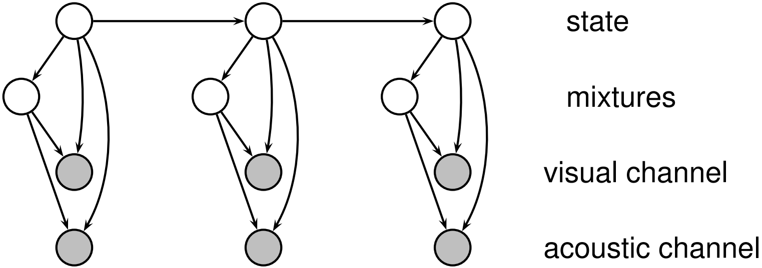
acoustic channel

visual channel

- no synchronisation at all

Dynamic Bayesian Networks

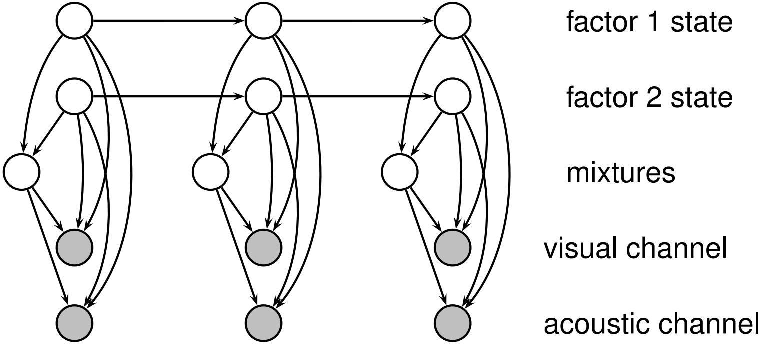
- product model



- state values are taken from the cross product of acoustic and visual states
- large probability distributions have to be trained

Dynamic Bayesian Networks

- factorial model (NEFIAN ET AL., EURASIP Journal on Applied Signal Processing, 2002(11))



- independent (hidden) states
- indirect influence by means of the ”explaining away” effect
- loose coupling of input channels

Dynamic Bayesian Networks

- inference is expensive
  - nodes are connected across slides
  - domains are not locally restricted
  - cliques became intractably large
- but: joint distribution usually need not be computed
  - only maximum detection required
  - Viterbi-like inference algorithms

Dynamic Bayesian Networks

- if computation of the joint probability is really required
  - partion the set of output variables O into $\{O_1, O_2, ..., O_n\}$ and instead of passing $P(O) = P(O_1, O_2, ..., P_n)$
  - pass $\{P(O_1), P(O_2), ..., P(O_n)\}$
  - error does not accumulate over time but converges to a finite error (Kullback-Leibler divergence)

Dynamic Bayesian Networks

- if space is bounded
  - recursive conditioning: trading space for time
    - instead of traversing the computation tree bottom-up and marginalizing out variables, the computation starts at the top node
    - space requirements is linear in the number of variables, but time requirements grow exponential
  - stochastic approximation: trading space for accuracy
    - simulation of likelihood estimation
    - to compute a $P(X)$ large numbers of configurations over the variables in the network are drawn using the conditional probabilities of the network

Bayesian Learning

- Bayesian Reasoning
- Bayes Optimal Classifier
- Naïve Bayes Classifier
- Cost-Sensitive Decisions
- Modelling with Probability Density Functions
- Parameter Estimation
- Bayesian Networks
- Markov Models
- Dynamic Bayesian Networks

Conditional Random Fields

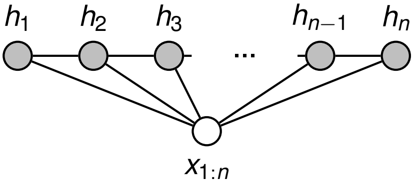
- hidden markov models ...
  - ... describe a joint probability distribution $p(x, h)$ over observation-label sequences
  - ... require a generative model of the domain: $p(x|h)$
    - enumerates all possible observation sequences x
    - generation not directly necessary for the task
  - ... make a simplifying assumption: observation depends only on the state of the model
- simplification only justified in some cases, usually
  - multiple interacting features
  - long range dependencies
- also: generative models are sometimes difficult to obtain

Conditional Random Fields

- partly contradictory goals
  - tractable inference and trainability $\rightarrow$ simple models
  - avoiding unwarranted independence assumptions $\rightarrow$ richer models
- reconciling the goals: direct learning of the probability $p(h|x)$
- $\rightarrow$ discriminative training
- no effort wasted on modelling the observations

Conditional Random Fields

- modelling the dependency of a set of variables on the whole input sequence
- undirected graphical model
- globally conditioned on the observation sequence x
- nodes in the graph correspond to random variables for elements in the label sequence
- Markov assumption: edges in the graph model the dependencies
- simplest model structure: chain of nodes



- in case of (potentially) infinite observations the variables are defined for a window of observations

Conditional Random Fields

- global probability distribution modelled as the normalized product of local potential functions
  - positive real valued functions
  - defined for subsets of the random variables
- Markov property: variables are conditionally independent given all the other variables in the model *if no edge exists between them*
  - potential functions defined over maximum cliques of the graph
  - only nodes which are directly connected are members of a maximum clique
  - for chains of nodes: potential functions operate on pairs of adjacent nodes (label variables) only

Conditional Random Fields

- isolated potential functions have no direct probabilistic interpretation
- represent constraints on the configuration of random variables over which the function is defined
  - local potential functions affect the global probability
  - a global configuration with a high probability is likely to satisfy more of these constraints than a configuration with a low probability

Conditional Random Fields

- potential functions have the form

$$\exp(\sum_j)\lambda_j t_j(h_{i-1}, h_i, x_{1:n}, i) + \sum_k \mu_k s_k(h_i, x_{1:n}, i)$$

  - $t_j(h_{i-1}, h_i, x_{1:n}, i)$: transition functions
  - $s_k(h_i, x_{1:n}, i)$: state function
  - $\lambda_{1:n}$ and $\mu_{1:n}$: parameters to be trained
- relationship to the observation: real valued feature functions, i.e.

$$b(x_{1:n}, i) = \begin{cases} 1 & \text{if } x_i = \text{ september} \\ 0 & \text{otherwise} \end{cases}$$

- transition functions defined in terms of feature functions, i.e.

$$t_j(h_{i-1}, h_i, x_{1:n}, i) = \begin{cases} b(x_{1:n}) & \text{if } y_{i-1} = \text{ IN } \wedge y_i = \text{ NNP} \\ 0 & \text{otherwise} \end{cases}$$

Conditional Random Fields

- global probability

$$p(y_{1:n}|x_{1:n}, \lambda_{1:n}) = \frac{1}{Z(x_{1:n})} \exp(\sum_j \lambda_j F_j(y_{1:n}, x_{1:n}))$$

with

$$F_j(y_{1:n}, x_{1:n}) = \sum_{i=1}^{n} f_i(h_{i-1}, h_i, x_{1:n}, i)$$

which are generalized transition and state functions
$Z(x_{1:n})$: normalizing factor

Conditional Random Fields

- motivated by the principle of maximum entropy
- maximum entropy: probability distribution should be as uniform as possible

principle of maximum entropy

The only probability distribution which can justifiably be constructed from incomplete data is the one which has maximum entropy subject to a set of constraints representing the given information.

- incomplete data: finite training set

Conditional Random Fields

- log-likelihood function of a conditional random field is concave
  $\rightarrow$ convergence to the global optimum is guaranteed
- usually no analytical solution for the maximum available
  $\rightarrow$ iterative approximation required

Conditional Random Fields

- for chain models probability can be computed as a sequence of matrix multiplications

$$M_i(h', h|x_{1:n}) = \exp(\sum_j \lambda_j f_j(h', h, x_{1:n}, i))$$

- $(n+1 \times n+1)$ matrices
  including reserved symbols for start and end of the sequence
- global probability

$$p(h_{1:n}|x_{1:n}, \lambda_{1:n}) = \frac{1}{Z(x_{1:n})} \prod_{i=1}^{n+1} M_i(h_{i-1}, h_i|x_{1:n})$$

- normalizing factor

$$Z(x_{1:n}) = \left[\prod_{i=1}^{n+1} M_i(x_{1:n})\right]_{\text{start,end}}$$

Conditional Random Fields

- training as dynamic programming
- forward and backward coefficients
  - similar to the hidden markov model case
  - but now vectors

$$\alpha_0(h|x_{1:n}) = \begin{cases} 1 & \text{if } h = \text{ start} \\ 1 & \textit{otherwise} \end{cases}$$

$$\beta_n + 1(h|x_{1:n}) = \begin{cases} 1 & \text{if } h = \text{ end} \\ 1 & \textit{otherwise} \end{cases}$$

$$\alpha_i(x_{1:n})^T = \alpha_{i-1}(x_{1:n})^T M_i(x_{1:n})$$

$$\beta_i(x_{1:n}) = M_{i+1}(x_{1:n})\beta_{i+1}(x_{1:n})$$

Conditional Random Fields

- e.g. probability of a transition from h' to h at time $i-1$ for a given training sequence $x^t_{1:n}$

$$p(h', h | x^t_{1:n}, \lambda_{1:n}) = \frac{\alpha_{i-1}(h' | x_{1:n}) M_i(h', h | x_{1:n}) \beta_i(h | x_{1:n})}{Z(x_{1:n})}$$