Uncertainty in Artificial Intelligence

Formularium exam

Tinne De Laet, Luc De Raedt

KU LEUVEN

2020-2021

marignal, conditional probability, Bayes' rule

Definition (marginalization)

given a joint distr. $p(x, y)$ the marginal distribution of x is defined by

$$p(x) = \sum_y p(x, y)$$

more generally,

$$p(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = \sum_{x_i} p(x_1, \ldots, x_n)$$

Definition (conditional probability)

$p(x|y) \equiv \frac{p(x,y)}{p(y)}$

Theorem (Bayes' rule)

$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$

independence

independence

Variables x and y are independent if knowing the state of one variable gives no extra information about the other variable:

$$p(x, y) = p(x)p(y)$$

If $p(x|y) = p(x)$ for all states of x and y, then the variables x and y are said to be independent.

Notation: $x \perp\!\!\!\perp y$.

interpretation

Note that $x \perp\!\!\!\perp y$ doesn't mean that there is no relation between x and y. It means that y contains no additional information on $p(x)$ (i.e. knowing y does not add information for $p(x)$ (with respect to knowing $p(x, y)$)).

factorisation

$p(x, y) = kf(x)g(y) \Rightarrow x \perp\!\!\!\perp y$.

prior, likelihood and posterior

the Max Likelihood assignment

The Max Likelihood (ML) setting is the one that maximises the likelihood,

$$\theta_* = \underset{\theta}{\operatorname{argmax}}\ p(\mathbf{d}|\theta, M)$$

The MAP assignment

the Most probable A Posteriori (MAP) estimate is the one that maximises the posterior,

$$\begin{aligned} \theta_* &= \underset{\theta}{\operatorname{argmax}}\ p(\theta|\mathbf{d}, M) = \underset{\theta}{\operatorname{argmax}}\ \frac{p(\mathbf{d}|\theta, M)p(\theta|M)}{p(\mathbf{d}|M)} && (1) \\ &= \underset{\theta}{\operatorname{argmax}}\ p(\mathbf{d}|\theta, M)p(\theta|M) && (2) \end{aligned}$$

belief networks (Bayesian networks)

belief network
A belief network is a DAG in which each node has associated the conditional probability of the node given its parents.

The joint distribution has <u>structured factorization</u>: product of the conditional probabilities, i.e.

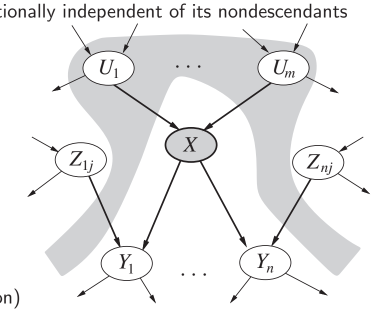
$$p(x_1, ..., x_D) = \prod_1^D p(x_i|pa(x_i))$$

Semantics

Factorization (= Global Semantics)
The joint factorizes as the product of the local conditional distributions

$$p(x_1, ..., x_D) = \prod_1^D p(x_i|pa(x_i))$$

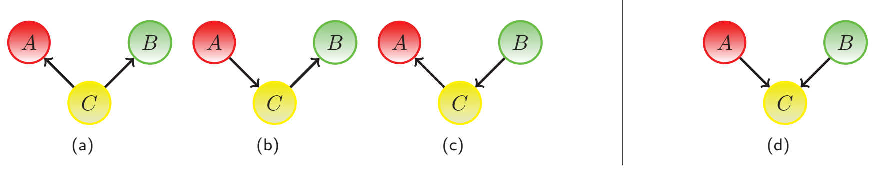
Local Semantics
Local semantics: each node is conditionally independent of its nondescendants given its parents (cf. also d-separation)



theorem
Local semantics $\leftrightarrow$ *global semantics*

conditional independence $\perp\!\!\!\perp$ in belief networks

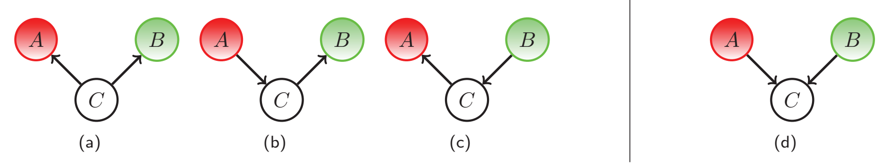
$A \perp\!\!\!\perp B \mid C$ $\quad$ $A \not\perp\!\!\!\perp B \mid C$



- In (a), (b) and (c), A, B are conditionally independent given C.
- In (d) the variables A, B are conditionally dependent given C, $p(A, B|C) \propto p(C|A, B)p(A)p(B)$.

(unconditional) independence $\perp\!\!\!\perp$ in belief networks

$A \not\perp\!\!\!\perp B$ $\quad$ $A \perp\!\!\!\perp B$



- In (a), (b) and (c), the variables A, B are marginally dependent.
- In (d) the variables A, B are marginally independent.

Distributional versus graphical independence !

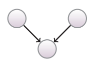
general rule for independence in belief networks

$\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{C}$

Given three sets of nodes $\mathcal{X}, \mathcal{Y}, \mathcal{C}$, if **all paths** from any element of $\mathcal{X}$ to any element of $\mathcal{Y}$ are **blocked** by $\mathcal{C}$, then $\mathcal{X} \perp\!\!\!\perp \mathcal{Y} | \mathcal{C}$.
We can also say that the any node in $\mathcal{X}$ **is d-seperated from any node in** $\mathcal{Y}$ **given all nodes in** $\mathcal{C}$**.**

blocked path

A path $\mathcal{P}$ is blocked by $\mathcal{C}$ if at least one of the following conditions is satisfied:

1. there is a collider  in the path $\mathcal{P}$ such that neither the collider nor any of its descendants is in the conditioning set $\mathcal{C}$.
2. there is a non-collider in the path $\mathcal{P}$ that is in the conditioning set $\mathcal{C}$.

Markov equivalence

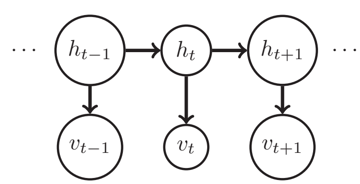
Markov Equivalence

Two graphs are **Markov equivalent** if they both represent the same set of conditional independence statements.

Hidden Markov Models (HMM)

This is a popular time series model used throughout many different fields (Machine Learning, Statistics, Tracking, Bioinformatics and many many more).

- A set of discrete or continuous variables $v_1, \ldots, v_T \equiv v_{1:T}$ which represent the observed time-series.
- A set of discrete hidden variables $h_{1:T}$ that generate the observations.



$$p(v_{1:T}, h_{1:T}) = p(v_1|h_1)p(h_1)\prod_{t=2}^{T} p(v_t|h_t)p(h_t|h_{t-1})$$

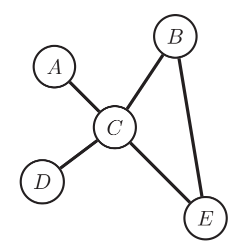
$p(h_t = j|h_{t-1} = i) = \pi_{ji}$, π: transition matrix
$p(v_t = j|h_t = i) = \rho_{ji}$, ρ: emission matrix
The past is independent of the future given the present !

Markov Network

Clique: Fully connected subset of nodes.

Maximal Clique: Clique which is not a subset of a larger clique.

A Markov Network is an undirected graph in which there is a potential (non-negative function) ψ defined on each (maximal) clique. (Maximality is not always assumed.)



The joint distribution is proportional to the product of all clique potentials.

$$p(A, B, C, D, E) = \frac{1}{Z}\psi(A, C)\psi(C, D)\psi(B, C, E)$$

$$Z = \sum_{A,B,C,D,E} \psi(A, C)\psi(C, D)\psi(B, C, E)$$

Markov Properties

Factorization in Markov Networks

$$p(x_1, ..., x_D) = \frac{1}{Z} \prod_{1}^{C} \phi_c(\mathcal{X}_c)$$

Local Markov Property – Neigbors – Markov blanket

$$p(x|\mathcal{X}\backslash x) = p(x|Ne(x))$$

with $Ne(x)$ the neighbors of x in the graph; When a distribution satisfies this property for all $x \in \mathcal{X}$ it is a **Markov Random Field**. w.r.t. graph G.

Pairwise Markov Property (follows from local property)

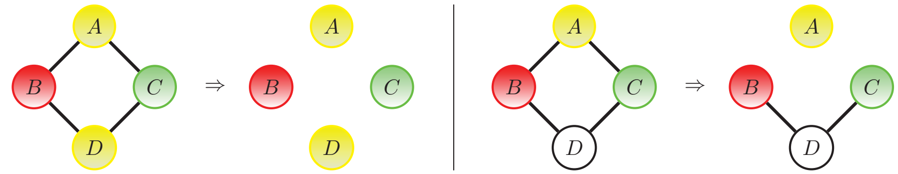
$$x \perp\!\!\!\perp y|\mathcal{X}\backslash\{x, y\}$$

when x and y are not adjacent.

Global Markov Property – see below

Property: These Markov Properties are equivalent if all potentials are *positive*.
Hammersley-Clifford Theorem If all potentials are positive, the Markov properties hold if and only if the distribution factorizes in the clique potentials.

General Rule for Independence in Markov Networks



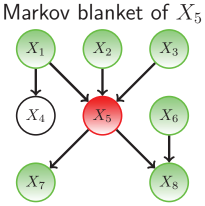
Global Markov Property

- Remove all links neighbouring the variables in the conditioning set $\mathcal{Z}$.
- If there is no path from any member of $\mathcal{X}$ to any member of $\mathcal{Y}$, then $\mathcal{X}$ and $\mathcal{Y}$ are conditionally independent given $\mathcal{Z}$.

Markov blanket

Remember: belief/bayesian networks

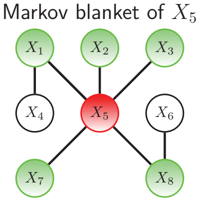
Markov blanket of X_5



Markov blanket: the Markov blanket of a node is its parents, children, and the parents of its children (but does not contain the node itself)!
property: giving/observing the Markov blanket renders a node independent of the other nodes in the graph

Now: **Markov networks** ...

Markov blanket of X_5



Markov blanket: the Markov blanket of a node is its neighbours
property: giving/observing the Markov blanket renders a node independent of the other nodes in the graph

Dependence, Independence, Perfect map

- For a distribution P we can write out a list $\mathcal{L}_P$ of all the independence statements.
- For a graph G, we can write a list of all the possible independence statements $\mathcal{L}_G$ implied by the semantics of the graph (Belief or Markov network).

Then we define:

$\mathcal{L}_P \subseteq \mathcal{L}_G$	Dependence Map (D-map)
$\mathcal{L}_P \supseteq \mathcal{L}_G$	Independence Map (I-map)
$\mathcal{L}_P = \mathcal{L}_G$	Perfect Map

In the above we assume the statement l is contained in $\mathcal{L}$ if it is consistent with (can be derived from) the independence statements in $\mathcal{L}$.

Inference rules for conditional independencies

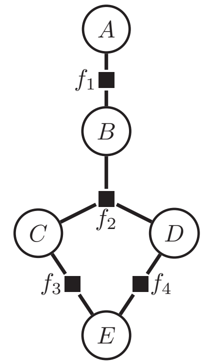
- Symmetry: $A \perp\!\!\!\perp B | C$ IMPLIES $B \perp\!\!\!\perp A | C$
- Decomposition: $A \perp\!\!\!\perp B, D$ IMPLIES $A \perp\!\!\!\perp B$ and $A \perp\!\!\!\perp D$
- Weak Union: $A \perp\!\!\!\perp B, D$ IMPLIES $A \perp\!\!\!\perp B | D$ and $A \perp\!\!\!\perp D | B$
- Contraction: $A \perp\!\!\!\perp B$ AND $A \perp\!\!\!\perp C | B$ IMPLIES $A \perp\!\!\!\perp B, C$

Notice that

- $A \perp\!\!\!\perp B | C$ DOES NOT IMPLY $A \perp\!\!\!\perp B | C, D$ (think of colliders)
- $A \perp\!\!\!\perp B$ DOES NOT IMPLY $A \perp\!\!\!\perp B | C$ (think of colliders)
- Because $A \perp\!\!\!\perp B$ IMPLIES $B \perp\!\!\!\perp A$, we also have for all K that $A \perp\!\!\!\perp B | K$ IMPLIES $B \perp\!\!\!\perp A | K$ (conditioning on both sides is allowed)

Factor Graphs

A square node represents a factor (non negative function) of its neighbouring variables.



The joint function is the product of all factors:

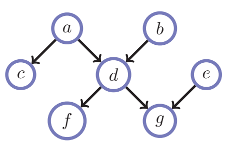
$$f(A, B, C, D, E) = f_1(A, B) f_2(B, C, D) f_3(C, E) f_4(D, E)$$

Factor graphs are useful for performing efficient computations (not just for probability).

variable elimination

calculate a marginal from a joint distribution
$\Rightarrow$ marginalize over all variables except marginal variables
e.g.

$$p(f) = \sum_{a,b,c,d,e,g} p(a, b, c, d, e, f, g)$$



(a)

$$p(f) = \sum_{a,b,c,d,e,g} p(a, b, c, d, e, f, g) = \sum_{a,b,c,d,e,g} p(f|d)p(g|d,e)p(c|a)p(d|a,b)p(a)p(b)p(e)$$

If we push the sumations inside in order e, c, b, g, a, d

$$p(f) = \sum_d p(f|d) \sum_a p(a) \sum_g \sum_b p(d|a,b)p(b) \sum_c p(c|a) \sum_e p(g|d,e)p(e)$$

bucket elimination

procedure

1. define an ordering of the variables beginning with "marginal variable"
2. draw the buckets starting with the "marginal variable" at the bottom
3. distribute the potentials over the buckets in the first column:
   - start with the highest bucket and put all potentials mentioning the variable (the bucket potentials)
   - go to the next bucket and put all REMAINING potentials mentioning the variable
   - ...
4. eliminate the buckets, by iterating:
   - go to the highest (non-marginalized) bucket and:
     - marginalize the product of the bucket potentials and the bucket messages over the bucket variable
     - send the result = (message) to the highest bucket with bucket variable present in the message
     - write non-eliminated potentials and the message in the next column
5. the product of the bucket potentials and bucket messages on the last row, last column is the marginal

sum and max product algorithm for factor graphs

In a tree exact inference of **all** the marginals can be done by two passes of the sum/max-product algorithm

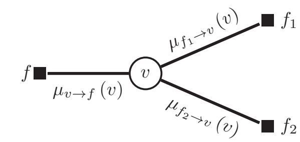
procedure

1. pick one node as the root node
2. initialize:
   - messages from leaf factor nodes initialized to factors
   - messages from leaf variable nodes set to unity
3. step 1: propagate messages from leaves to root
4. step 2: propagate messages (or backtrack for max-product) from root to leaves

sum-product algorithm for factor graphs

variable to factor message

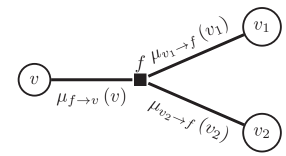
$$\mu_{v\to f}(v) = \prod_{f_i \sim v \setminus f} \mu_{f_i \to v}(v)$$



messages from extremal variables are set to 1

factor to variable message

$$\mu_{f\to v}(v) = \sum_{\{v_i\}} f(v, \{v_i\}) \prod_{v_i \sim f \setminus v} \mu_{v_i \to f}(v_i)$$



messages from extremal factors are set to the factor

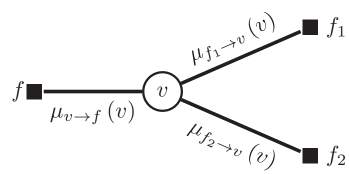
Marginal

$p(v) \propto \prod_{f_i \sim v} \mu_{f_i \to v}(v)$

max-product algorithm for factor graphs

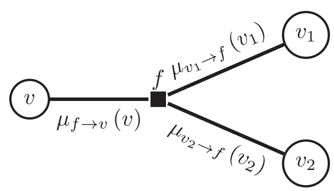
variable to factor message

$$\mu_{v\to f}(v) = \prod_{f_i \sim v \setminus f} \mu_{f_i \to v}(v)$$



factor to variable message

$$\mu_{f\to v}(v) = \max_{\{v_i\}} f(v, \{v_i\}) \prod_{v_i \sim f \setminus v} \mu_{v_i \to f}(v_i)$$



marginal

$v^* = \underset{v}{\operatorname{argmax}} \prod_{f_i \sim v} \mu_{f_i \to v}(v)$

Learning settings

always keep in mind whether we learn

- the parameters of the graphical model ?
  - point estimates (ML, MAP) (maximum likelihood, maximum a posteriori)
  - a distribution over these parameters (from prior to posterior) $\to$ bayesian approach.
- the structure of the graphical model ?

always keep in mind what data is available

Is the data fully or partially observable ?

Prior, Likelihood and Posterior

More fully, if we condition on the model M, we have

$$p(\theta|\mathbf{d}, M) = \frac{p(\mathbf{d}|\theta, M)p(\theta|M)}{\underbrace{p(\mathbf{d}|M)}_{\text{marginal likelihood}}}$$

The MAP assignment (mode of posterior)
$\theta_* = \underset{\theta}{\operatorname{argmax}}\ p(\theta|\mathbf{d}, M)$

The Max Likelihood assignment (mode of likelihood)
$\theta_* = \underset{\theta}{\operatorname{argmax}}\ p(\mathbf{d}|\theta, M)$

Remark:
if $p(\theta|M) =$ const. $\Rightarrow$ MAP=ML
$\theta_* = \underset{\theta}{\operatorname{argmax}}\ p(\theta|\mathbf{d}, M) = \underset{\theta}{\operatorname{argmax}}\ \frac{p(\mathbf{d}|\theta,M)p(\theta|M)}{p(\mathbf{d}|M)} = \underset{\theta}{\operatorname{argmax}}\ p(\mathbf{d}|\theta, M)$

Point estimate, full distribution

1. **Full distribution:** posterior $p(\theta|\mathbf{d}, M)$
2. **Point estimates:**
   1. Maximum Likelihood (ML):
      $$\theta_* = \underset{\theta}{\operatorname{argmax}}\ p(\mathbf{d}|\theta, M)$$
   2. Maximum a posterior (MAP):
      $$\theta_* = \underset{\theta}{\operatorname{argmax}}\ p(\theta|\mathbf{d}, M)$$
   3. Mean of posterior: $< p(\theta|\mathbf{d}, M) >$

Usually one makes the i.i.d assumption (identically and independently distributed):

$$p(x^1, ..., x^n \mid \theta) = \prod_i p(x^i \mid \theta)$$

BN parameter learning: MAP & beta priors

$$p(\theta_a) = B\left(\theta_a|\alpha_a, \beta_a\right) = \frac{1}{B(\alpha_a, \beta_a)}\theta_a^{\alpha_a - 1}\left(1 - \theta_a\right)^{\beta_a - 1}$$

for which the posterior is also a Beta distribution:

$$p(\theta_a|\mathcal{V}_a) = B\left(\theta_a|\alpha_a + \sharp\left(a = 1\right), \beta_a + \sharp\left(a = 0\right)\right)$$

The marginal table is given by

$$p(a = 1|\mathcal{V}_a) = \int_{\theta_a} p(\theta_a|\mathcal{V}_a)\theta_a = \mathbb{E}\left[\theta_a\right] = \frac{\alpha_a + \sharp\left(a = 1\right)}{\alpha_a + \sharp\left(a = 1\right) + \beta_a + \sharp\left(a = 0\right)}$$

---

hyperparameters
The prior parameters α_a, β_a are called hyperparameters. If one had no preference, one would set $\alpha_a = \beta_a = 1$.

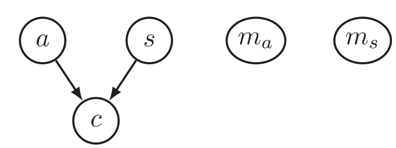
Missing Completely at random (MCAR)

Notation $\mathbf{x}, \mathbf{y}$ for sets of variables, $\mathbf{m}$ for missingness variables,
x for single variable.

MCAR
MCAR states that $\mathbf{x} \perp\!\!\!\perp \mathbf{m}$
or for some variable x we have $x, \mathbf{y} \perp\!\!\!\perp \mathbf{m}$

therefore

$$\begin{aligned}
\Pr(x, \mathbf{y}, \mathbf{m}) &= \Pr(x, \mathbf{y}) \cdot \Pr(\mathbf{m}) \\
\Pr(x, \mathbf{y}, \mathbf{m} = \text{ob}) &= \Pr(x, \mathbf{y}) \cdot \Pr(\mathbf{m} = \text{ob}) \\
\Pr(x, \mathbf{y}) &= \Pr(x, \mathbf{y}|\mathbf{m} = \text{ob}) \\
\Pr(x|\mathbf{y}) &= \Pr(x|\mathbf{y}, \mathbf{m} = \text{ob}).
\end{aligned}$$



Missingness graph for MCAR.

Therefore estimate $\Pr(x|\mathbf{y}) = \Pr(x|\mathbf{y}, \mathbf{m} = \text{ob})$.

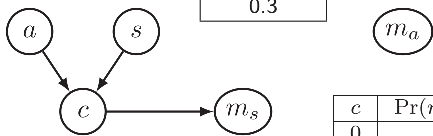
Missing at random

MAR
This assumption holds if the missingness variables are conditionally independent of the partially-observed variables x_m given the fully-observed variables x_o, i.e., if $x_m \perp\!\!\!\perp m \mid x_o$. Graphically, this corresponds to a missingness graph where the $\mathbf{m}$ are only allowed to have parents that are fully observed.

$\Pr(a=1)$
0.1

$\Pr(s=1)$
0.3

$\Pr(m_a = \mathsf{ob})$
0.5



a	s	$\Pr(c=1\|a,s)$
0	0	0.1
0	1	0.3
1	0	0.2
1	1	0.4

c	$\Pr(m_s = \mathsf{ob}\|c)$
0	0.9
1	0.2

Maximum Likelihood with missing data

Likelihood for complete data

$$\mathcal{L}(\theta|\boldsymbol{\mathcal{D}}) = \Pr(\boldsymbol{\mathcal{D}}|\theta) = \prod_{i=1}^{N} \Pr(\mathbf{d}_i|\theta)$$

where N is the dataset size and $\mathbf{d}_i$ represents the assignments made in the ith complete example of the dataset.

Marginal Likelihood (for partially observed data)

$$\mathcal{L}(\theta|\boldsymbol{\mathcal{D}}) = \Pr(\boldsymbol{\mathcal{D}}|\theta) = \prod_{i=1}^{N} \Pr(\mathbf{d}_i|\theta) = \prod_{i=1}^{N} \sum_{\mathbf{h}_i} \Pr(\mathbf{d}_i, \mathbf{h}_i|\theta).$$

with $\mathbf{h}_i$ the hidden variables in example i.
Global and local parameter independence (as for fully observed data) does not hold any more !
The parameters are coupled, a different optimisation method is needed.

The classical EM algorithm

- **Given**: joint distribution $p(\mathbf{d}, \mathbf{h}|\theta)$ (model) over visible/observed variables v and hidden variables h, governed by parameters θ
- **Goal:** find $\theta_{ML} = \underset{\theta}{\operatorname{argmax}} \ \log p(\boldsymbol{\mathcal{D}}|\theta) = \underset{\theta}{\operatorname{argmax}} \ \log \left\{\sum_{\mathbf{h}} p(\boldsymbol{\mathcal{D}}, \mathbf{h}|\theta)\right\}$
- **Procedure:**
  1. choose initial setting for parameters θ^{old}
  2. **E-step:** evaluate $p(\mathbf{h}|\boldsymbol{\mathcal{D}}, \theta^{old})$ (posterior over missing variables)
  3. **M-step:** find $\theta^{new} = \underset{\theta}{\operatorname{argmax}} \ \mathcal{Q}\left(\theta, \theta^{old}\right)$ (maximise expected complete data log likelihood)
     with $\mathcal{Q}\left(\theta, \theta^{old}\right) = \sum_{\mathbf{h}} \underbrace{p(h|\boldsymbol{\mathcal{D}}, \theta^{old})}_{\text{posterior over missing variables}} \underbrace{\log\{p(\boldsymbol{\mathcal{D}}, \mathbf{h}|\theta)\}}_{\text{complete data log likelihood}}$
  4. if not converged $\theta^{old} = \theta^{new}$ and return to step 2.

Univariate sampling: discrete distributions

Consider the one dimensional discrete distribution $p(x)$ where $\mathrm{dom}(x) = \{1, 2, 3\}$, with

$$p(x) = \begin{cases} 0.6 & x = 1 \\ 0.1 & x = 2 \\ 0.3 & x = 3 \end{cases}$$

Create cumulant:

$$c(x) = \begin{cases} c_0 & 0 \\ c_1 & p(x=1) = 0.6 \\ c_2 & p(x=1) + p(x=2) = 0.7 \\ c_3 & p(x=1) + p(x=2) + p(x=3) = 1.0 \end{cases}$$

1 $\times$	2	3

We then draw a sample uniformly from $[0, 1]$, say $u = 0.47$. Then the sampled state would be state 1, since this is in the interval $(c_0, c_1]$.

Univariate sampling: continuous distributions

- First we calculate the cumulant density function

$$C(y) = \int_{-\infty}^{y} p(x)dx$$

- Then we sample u uniformly from $[0, 1]$, and obtain the corresponding sample x by solving $C(x) = u \Rightarrow x = C^{-1}(u)$.

Special cases
For certain distributions, such as Gaussians, numerically efficient alternative procedures exist, usually based on co-ordinate transformations.

Rejection Sampling

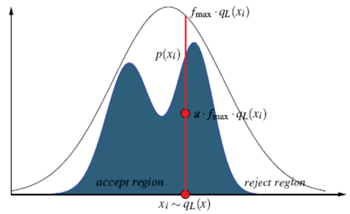
preparation

- look for a proposal $q(x)$
  - that we can sample from
  - that when scaled with factor M such that $\forall x : p^*(x) \leq M\ q(x))$

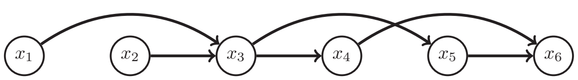
rejection sampling

- draw a candidate x^{cand} from $q(x)$ (normal distribution)
- draw a value u uniformly between 0 and 1.
- calculate $a = \frac{p^*(x^{cand})}{Mq(x^{cand})}$

$$\begin{cases} \text{if } u < a & \text{accept } x^{cand} \\ \text{else} & \text{reject } x^{cand} \end{cases}$$



Ancestral Sampling for Belief Nets



- We first rename the variable indices so that parent variables always come before their children

$$p(x_1, \ldots, x_6) = p(x_1)p(x_2)p(x_3|x_1, x_2)p(x_4|x_3)p(x_5|x_3)p(x_6|x_4, x_5)$$

- 1. Sample **from those nodes that do not have any parents** (here x_1 and x_2)
  2. Sample **nodes of which you already sampled the parents**, i.e sample x_i from $p(x_i|pa(x_i))$
  3. iterate

  In example: given samples for x_1 and x_2 sample x_3, and then x_4 and x_5 and finally x_6.
- Despite loops ancestral sampling is straightforward (both discrete and continuous variables) if you can sample from all $p(x_i|pa(x_i))$
- Ancestral or 'forward' sampling is a case of **perfect sampling**

Ancestral sampling with evidence

- Consider sampling from $p(x_1, x_2, x_3, x_4, x_5|x_6)$. Using Bayes' rule, this is

$$\frac{p(x_1)p(x_2)p(x_3|x_1, x_2)p(x_4|x_3)p(x_5|x_3)p(x_6|x_4, x_5)}{\sum_{x_1,x_2,x_3,x_4,x_5} p(x_1)p(x_2)p(x_3|x_1, x_2)p(x_4|x_3)p(x_5|x_3)p(x_6|x_4, x_5)}$$

  Now x_4 and x_5 are coupled. One could attempt to work out an equivalent new forward sampling structure, although generally this will be as complex as running an exact inference approach.
- An alternative is to proceed with forward sampling from the non-evidential distribution, and discard any samples which do not match the evidential states. This is generally not recommended since the probability that a sample will be consistent with the evidence is typically very small.

Gibbs Sampling

- We want to generate sample from $p(x_1, x_2, \ldots, x_n)$
- Start with an initial sample set (e.g. just pick one) $x^0 = (x_1^0, x_2^0, \ldots, x_n^0)$
- Choose a particular variable, x_i, for which we draw a new sample
  Factorize joint as:

$$p(x) = p(x_i | x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) p(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$$

- Given the joint initial state $x^0 = \left(x_1^0, \ldots, \ldots, x_n^0\right)$ draw a sample x_i^1 from

$$p(x_i | x_1^0, \ldots, x_{i-1}^0, x_{i+1}^0, \ldots, x_n^0) \equiv p(x_i | x_{\setminus i})$$

  We assume this distribution is easy to sample from since it is univariate.
  $\rightarrow$ new joint sample (in which only x_i has been updated)
  $x^1 = \left(x_1^0, \ldots, x_{i-1}^0, x_i^1, x_{i+1}^0, \ldots, x_n^1\right)$.
- Then select another variable x_j to sample $\rightarrow$
  $x^2 = \left(x_1^1, \ldots, x_{j-1}^1, x_j^2, x_{j+1}^1, \ldots, x_n^1\right)$.
- Continue this procedure to obtain a set $x^1, \ldots, x^L$ of samples in which each x^{l+1} differs from x^l in only a single component.

Importance Sampling

Importance Sampling is a misnomer

- it is a technique to **approximate averages** with respect to an intractable distribution $p(x)$ **using samples** $\mathcal{X}$ **from 'another' distribution** $q(x)$
- it is **NOT a sampling technique**

- We want to calculate average of $f(x)$ with respect to $p(x)$ (for instance expected value, covariance, ...)
- $p(x) = \frac{p^*(x)}{Z}$ where $p^*(x)$ can be evaluated ($Z = \int_x p^*(x)$ = normalization constant)
- $\mathcal{X} = \left[x^1, \ldots, x^L\right]$ samples from $q(x)$
- average can be approximated by: $\int_x f(x)p(x) \approx \sum_{l=1}^L f(x^l) w^l$ with importance weights $w^l = \frac{p^*(x^l)/q(x^l)}{\sum_{l=1}^L p^*(x^l)/q(x^l)}$, with $\sum_{l=1}^L w^l = 1$

hidden Markov models: filtering

forward algorithm

- **goal:** $P(h_t \mid v_{1:t})$
- **underlying idea:** The forward algorithm provides a recursion for $p(h_t, v_{1:t}) = \alpha(h_t)$
- **alpha recursion:** $p(h_t, v_{1:t}) = \alpha(h_t) = \underbrace{p(v_t | h_t)}_{\text{corrector}} \underbrace{\sum_{h_{t-1}} p(h_t | h_{t-1}) \alpha(h_{t-1})}_{\text{predictor}}$

  with $\alpha(h_1) = p(h_1, v_1) = p(v_1 | h_1) p(h_1)$
- **filtering distribution** is obtained by normalization
  $P(h_t \mid v_{1:t}) = \frac{p(h_1, v_{1:t})}{p(v_{1:t})} \propto \alpha(h_t)$

hidden Markov models: smoothing

parallel smoothing = forward-backward algorithm = alpha-betarecursion

- $P(h_t, v_{1:T}) = \alpha(h_t)\beta(h_t)$
- $\alpha(h_t)$ from alpha-recursion (forward)
- $\beta(h_t)$ from beta-recursion (backward)

$$\begin{aligned}
\beta(h_{t-1}) &= P(v_{t:T} \mid h_{t-1}) \\
&= \sum_{h_t} P(v_t \mid h_t) P(h_t \mid h_{t-1}) \underbrace{P(v_{t+1:T} \mid h_t)}_{\beta(h_t)}
\end{aligned}$$

with $\beta(h_T) = 1$

hidden Markov models: most likely joint path

Viterbi-algorithm

- the most likely path $h_{1:T}$ of $p(h_{1:T}|v_{1:T})$ is the same as the most likely state of $P(h_{1:T}, v_{1:T})$
- **IDEA:** send message from end of the chain forward:

$$\mu(h_{t-1}) = \max_{h_t} P(v_t \mid h_t)\, P(h_t \mid h_{t-1})\, \mu(h_t) \tag{3}$$

  with $\mu(h_T) = 1$
- **BACKTRACING:** effect of maximizing over $h_2, \ldots, h_T$ is compressed into $\mu(h_1)$: $h_1^* = \underset{h_1}{\operatorname{argmax}}\ P(v_1 \mid h_1)\, P(h_1)\, \mu(h_1)$
  and: $h_t^* = \underset{h_t}{\operatorname{argmax}}\ P(v_t \mid h_t)\, P\big(h_t \bigm| h_{t-1}^*\big)\, \mu(h_t)$

a general Bayes filter algorithm

$\underline{\textbf{Algorithm Bayes\_filter}}(P(x_{t-1} \mid u_{1:t-1}, y_{1:t-1}))$
for all x_t **do**

$P(x_t \mid u_{1:t}, y_{1:t-1}) = \int P(x_t \mid u_t, x_{t-1})\, P(x_{t-1} \mid u_{t-1}, y_{1:t-1})\, dx_{t-1}$
$\Rightarrow$ prediction
$P(x_t \mid u_{1:t}, y_{1:t}) \propto P(y_t|x_t)\, P(x_t \mid u_{1:t}, y_{1:t-1})$
$\Rightarrow$ correction

endfor
return $P(x_t \mid u_{1:t}, y_{1:t})$

remark: prior $P(x_0)$ needed in first timestep

Kalman filter

Algorithm Kalman_filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, y_t)$
$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t \qquad \Rightarrow$ prediction
$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + Q_t$
$K_t = \bar{\Sigma}_t H_t^T \left(H_t \bar{\Sigma}_t H_t^T + R_t\right)^{-1}$
$\mu_t = \bar{\mu}_t + K_t\,(y_t - H_t \bar{\mu}_t) \Rightarrow$ correction
$\Sigma_t = (I - K_t H_t)\, \bar{\Sigma}_t$
return μ_t, Σ_t

bootstrap filter = most simple particle filter

Algorithm Bootstrap_filter (X_{t-1}, u_t, y_t)
$\bar{X}_t = X_t = \text{empty}$
for $m - 1$ to M do
$\underbrace{\text{sample } x_t^m \sim p(x_t|u_t, x_{t-1}^m)}_{\text{prediction}}$
$\underbrace{w_t^m = p(y_t|x_t^m)}_{\text{correction}}$
$\bar{X}_t = \bar{X}_t + < x_t^m, w_t^m >$
endfor
resample
for $m - 1$ to M do
draw i with probability $\propto w_t^i$
add x_t^i to X_t
endfor
return X_t

Satisfiability

Given

- a set of boolean variables (the numbers 1,..,9 in the game)
- a set of clauses of the form $l_1 \vee ... \vee l_k$ (k=3 in the game, 3-SAT), with each literal a positive or negative variable

Find: an assignment of truth-values to the variables such that all clauses are satisfied

The DPLL Algorithm for SAT

procedure $DPLL(Vars:$ variables, $S:$ set of clauses):
  if S is empty
    return 1
  else if S contains an empty clause
    return 0
  else select $v \in Vars$
    $S_t := S$ where $v = 1$ (making the variable true)
    $S_f := S$ where $v = 0$ (making the variable false)
    return $DPLL(Vars - \{v\}, S_t) + DPLL(Vars - \{v\}, S_f)$

- In a CNF theory $(A \vee \neg B) \wedge (C \vee D)$, the clauses are the disjunctions, that is, $(A \vee \neg B)$ and $(C \vee D)$
- A unit clause contains exactly one literal. E.g., A and $\neg A$ are both unit clauses. (It is possible to make DPLL more efficient by assigning unit clauses the appropriate value)
- An empty clause is a disjunction of 0 literals, at least one of which must be true. Therefore an empty clause is always unsatisfiable.

Algebraic Model Counting

- commutative semiring $(A, \oplus, \otimes, e^{\oplus}, e^{\otimes})$
- algebraic literals
  $L(F) = \{f_1, \ldots, f_n\} \cup \{\neg f_1, \ldots, \neg f_n\}$
- labeling function $\alpha : L(F) \rightarrow A$
- propositional logical theory T

$$AMC(T) = \bigoplus_{T \models w} \bigotimes_{l \in w} \alpha(l)$$

Useful Semirings

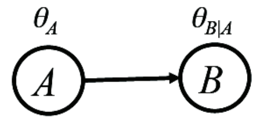
task	$\mathcal{A}$	$e^{\oplus}$	$e^{\otimes}$	$\oplus$	$\otimes$	$\alpha(v)$	$\alpha(\neg v)$	ref
SAT	$\{true, false\}$	$false$	$true$	$\vee$	$\wedge$	$true$	$true$	B, BT, G, GK, K, L, M
#SAT	$\mathbb{N}$	0	1	$+$	$\cdot$	1	1	B, G, GK, K, L
WMC	$\mathbb{R}_{\geq 0}$	0	1	$+$	$\cdot$	$\in \mathbb{R}_{\geq 0}$	$\in \mathbb{R}_{\geq 0}$	
PROB	$\mathbb{R}_{\geq 0}$	0	1	$+$	$\cdot$	$\in [0,1]$	$1 - \alpha(v)$	B, BT, E, G, K
SENS	$\mathbb{R}[\mathcal{V}]$	0	1	$+$	$\cdot$	v or $\in [0,1]$	$1 - \alpha(v)$	K
GRAD	$\mathbb{R}_{\geq 0} \times \mathbb{R}$	$(0,0)$	$(1,0)$	Eq. (4)	Eq. (5)	Eq. (2)	Eq. (3)	E, K
MPE	$\mathbb{R}_{\geq 0}$	0	1	max	$\cdot$	$\in [0,1]$	$1 - \alpha(v)$	B, BT, G, K, L, M
S-PATH	$\mathbb{N}^{\infty}$	∞	0	min	$+$	$\in \mathbb{N}$	0	BT, GK, K
W-PATH	$\mathbb{N}^{\infty}$	0	∞	max	min	$\in \mathbb{N}$	∞	BT
FUZZY	$[0,1]$	0	1	max	min	$\in [0,1]$	1	GK, M
kWEIGHT	$\{0, \ldots, k\}$	k	0	min	$+^k$	$\in \{0, \ldots, k\}$	$\in \{0, \ldots, k\}$	M
$\mathtt{OBDD}_<$	$\mathtt{OBDD}_<(\mathcal{V})$	$\mathtt{OBDD}_<(0)$	$\mathtt{OBDD}_<(1)$	$\vee$	$\wedge$	$\mathtt{OBDD}_<(v)$	$\neg\mathtt{OBDD}_<(v)$	K
WHY	$\mathcal{P}(\mathcal{V})$	$\emptyset$	$\emptyset$	$\cup$	$\cup$	$\{v\}$	n/a	GK
$\mathcal{RA}^+$	$\mathbb{N}[\mathcal{V}]$	0	1	$+$	$\cdot$	v	n/a	GK

Table 1: Examples of commutative semirings and labeling functions. The **WHY** and $\mathcal{RA}^+$ provenance semirings apply to positive literals only. Reference key: B (Bacchus et al., 2009), BT (Baras and Theodorakopoulos, 2010), E (Eisner, 2002), G (Goodman, 1999), GK (Green et al., 2007), K (Kimmig et al., 2011), L (Larrosa et al., 2010), M (Meseguer et al., 2006); more examples can be found in these references.

From Kimmig, Vanden Broeck and De Raedt, 2016

Encode a BN in WMC

A	Θ_A
true	$\theta_A = .3$
false	$\theta_{\neg A} = .7$



$$A \Leftrightarrow \theta_A \qquad \neg A \Leftrightarrow \theta_{\neg A}$$

$$A \wedge B \Leftrightarrow \theta_{B|A} \qquad A \wedge \neg B \Leftrightarrow \theta_{\neg B|A} \qquad \neg A \wedge B \Leftrightarrow \theta_{B|\neg A} \qquad \neg A \wedge \neg B \Leftrightarrow \theta_{\neg B|\neg A}$$

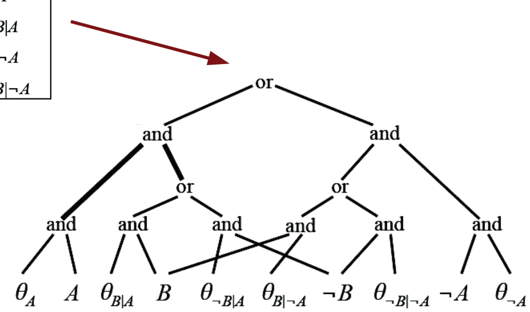
Encode a BN in WMC

- we start from the full CPT (with ALL entries, that is for both true and false)
- each possible entry/parameter in a CPT is encoded as a logical expression
- the weights are as follows: $w(A) = w(\neg A) = w(B) = w(\neg B) = 1$ and $w(\neg\theta_x) = 1$ but $w(\theta_x) =$ entry in CPT (as there is no evidence and we are interested in the weighted model count of the complete theory)

Knowledge Compilation

$$A \Leftrightarrow \theta_A \qquad \neg A \Leftrightarrow \theta_{\neg A}$$

$$A \wedge B \Leftrightarrow \theta_{B|A} \qquad A \wedge \neg B \Leftrightarrow \theta_{\neg B|A} \qquad \neg A \wedge B \Leftrightarrow \theta_{B|\neg A} \qquad \neg A \wedge \neg B \Leftrightarrow \theta_{\neg B|\neg A}$$



$$\{(\theta_A \wedge A) \wedge ((\theta_{B|A} \wedge B) \vee (\theta_{\neg B|A} \wedge \neg B))\} \vee \{(\theta_{\neg A} \wedge \neg A) \wedge ((\theta_{B|\neg A} \wedge B) \vee (\theta_{\neg B|\neg A} \wedge \neg B))\}$$

Types of Circuits

Negation normal form (NNF):

- only AND and OR allowed, furthermore leafs are literals (variables or their negation)
- we shall interpret this as logical circuits ...
- fill in values for the variables and compute whether the assignment satisfies the circuit or not.

NNFs : special forms

- Why is this important ?
  - remember that we replace and by x and or by +
- decomposability allows to rewrite $P(A \wedge B) = P(A) \times P(B)$
- determinism allows to rewrite $P(A \vee B) = P(A) + P(B)$
- without smoothness you might not take into account all variables
- these eqs. do not hold for arbitrary formula A and B !

NNFs and Decision Nodes

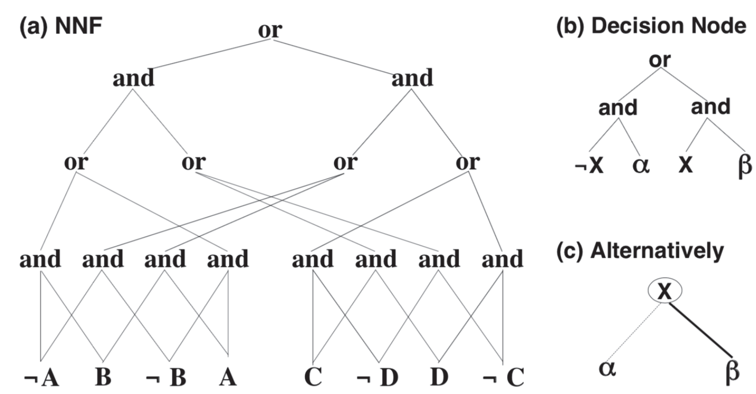


Figure 1: An NNF circuit and a decision node.

from [Huang & Darwiche, JAIR 2007]

Probabilistic Logic Programs

as in the probabilistic programming language ProbLog

Propositional logic program

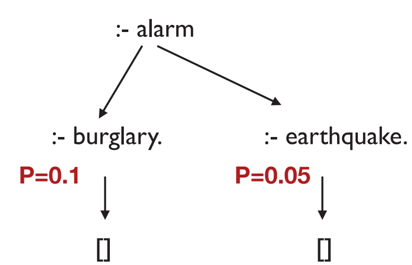
0.1 :: burglary.
0.3 ::hears_alarm(mary).

0.05 ::earthquake.
0.6 ::hears_alarm(john).

alarm :– earthquake.
alarm :– burglary.

calls(mary) :– alarm, hears_alarm(mary).
calls(john) :– alarm, hears_alarm(john).

Disjoint sum problem



Probability of one proof : $\prod_{f:fact \in Proof} P_f$

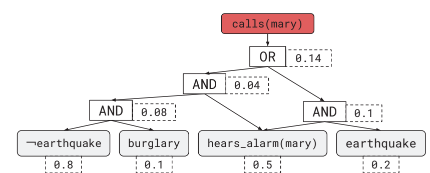
P(alarm) = P(burg OR earth)
= P(burg) + P(earth) - P(burg AND earth)
=/= P(burg) + P(earth)

ProbLog Inference

Answering a query in a ProbLog program happens in four steps

1. Grounding the program w.r.t. the query
2. **Rewrite the ground logic program into a propositional logic formula**
3. Compile the formula into an arithmetic circuit
4. Evaluate the arithmetic circuit

0.1 :: burglary.
0.5 :: hears_alarm(mary).
0.2 :: earthquake.
0.4 :: hears_alarm(john).
alarm :– earthquake.
alarm :– burglary.
calls(mary) :– alarm, hears_alarm(mary).
calls(john) :– alarm, hears_alarm(john).



calls(mary)

↔

hears_alarm(mary) ∧ (burglary ∨ earthquake)