INFERENCE

Seminar: Probabilistic Graphical Models

Johannes Langer

January 16, 2016

Department of Statistics

TABLE OF CONTENTS

FACTORS

Definition A distribution $P(\mathcal{X}) := P(X_1, ..., X_n)$ factorizes over a *Bayesian Network* $\mathcal{G}$, if P can be expressed as a product:

$$P(X_1, ..., X_n)_{\mathcal{G}} = \prod_{k=1}^{n} P_k(X_k | Pa_{X_k}^{\mathcal{G}}) \stackrel{as\ factor}{=} \prod_{k=1}^{n} \Phi_k(X_k, Pa_{X_k}^{\mathcal{G}}) \tag{1}$$

Definition A distribution $P(\mathcal{X}) := P(X_1, ..., X_n)$ factorizes over a *Bayesian Network* $\mathcal{G}$, if P can be expressed as a product:

$$P(X_1, ..., X_n)_{\mathcal{G}} = \prod_{k=1}^{n} P_k(X_k | Pa_{X_k}^{\mathcal{G}}) \stackrel{as\ factor}{=} \prod_{k=1}^{n} \Phi_k(X_k, Pa_{X_k}^{\mathcal{G}}) \tag{1}$$

Definition A distribution $P(\mathcal{X})$ factorizes over a *Markov Network* $\mathcal{H}$, if each $\mathbf{D_k} \subset \mathcal{X}$ $k \in (1, ...K)$ is a complete subgraph (clique) of $\mathcal{H}$:

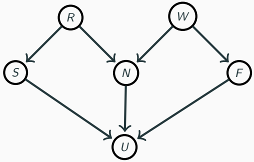
$$P(X_1, ..., X_n)_{\mathcal{H}} = \frac{1}{Z}\Phi(X_1, ..., X_n) \propto \prod_{k=1}^{K} \Phi_k(\mathbf{D_k}) \tag{2}$$

Definition A distribution $P(\mathcal{X}) := P(X_1, ..., X_n)$ factorizes over a *Bayesian Network* $\mathcal{G}$, if P can be expressed as a product:

$$P(X_1, ..., X_n)_{\mathcal{G}} = \prod_{k=1}^{n} P_k(X_k | Pa_{X_k}^{\mathcal{G}}) \overset{as\ factor}{=} \prod_{k=1}^{n} \Phi_k(X_k, Pa_{X_k}^{\mathcal{G}}) \tag{1}$$

Definition A distribution $P(\mathcal{X})$ factorizes over a *Markov Network* $\mathcal{H}$, if each $\mathbf{D_k} \subset \mathcal{X}$ $k \in (1, ...K)$ is a complete subgraph (clique) of $\mathcal{H}$:

$$P(X_1, ..., X_n)_{\mathcal{H}} = \frac{1}{Z}\Phi(X_1, ..., X_n) \propto \prod_{k=1}^{K} \Phi_k(\mathbf{D_k}) \tag{2}$$

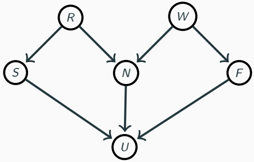


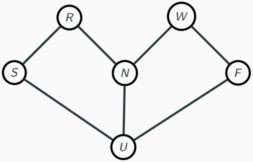
Definition A distribution $P(\mathcal{X}) := P(X_1, ..., X_n)$ factorizes over a *Bayesian Network* $\mathcal{G}$, if P can be expressed as a product:

$$P(X_1, ..., X_n)_{\mathcal{G}} = \prod_{k=1}^{n} P_k(X_k | Pa_{X_k}^{\mathcal{G}}) \stackrel{\text{as factor}}{=} \prod_{k=1}^{n} \Phi_k(X_k, Pa_{X_k}^{\mathcal{G}}) \tag{1}$$

Definition A distribution $P(\mathcal{X})$ factorizes over a *Markov Network* $\mathcal{H}$, if each $\mathbf{D_k} \subset \mathcal{X}$ $k \in (1, ...K)$ is a complete subgraph (clique) of $\mathcal{H}$:

$$P(X_1, ..., X_n)_{\mathcal{H}} = \frac{1}{Z} \Phi(X_1, ..., X_n) \propto \prod_{k=1}^{K} \Phi_k(\mathbf{D_k}) \tag{2}$$



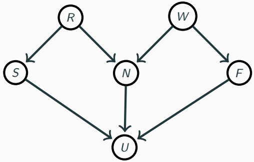


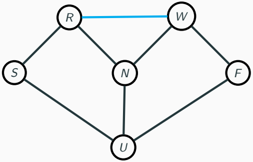
Definition A distribution $P(\mathcal{X}) := P(X_1, ..., X_n)$ factorizes over a *Bayesian Network* $\mathcal{G}$, if P can be expressed as a product:

$$P(X_1, ..., X_n)_{\mathcal{G}} = \prod_{k=1}^{n} P_k(X_k | Pa_{X_k}^{\mathcal{G}}) \overset{\text{as factor}}{=} \prod_{k=1}^{n} \Phi_k(X_k, Pa_{X_k}^{\mathcal{G}}) \tag{3}$$

Definition A distribution $P(\mathcal{X})$ factorizes over a *Markov Network* $\mathcal{H}$, if each $\mathbf{D_k} \subset \mathcal{X}$ $k \in (1, ...K)$ is a complete subgraph (clique) of $\mathcal{H}$:

$$P(X_1, ..., X_n)_{\mathcal{H}} = \frac{1}{Z}\Phi(X_1, ..., X_n) \propto \prod_{k=1}^{K} \Phi_k(\mathbf{D_k}) \tag{4}$$



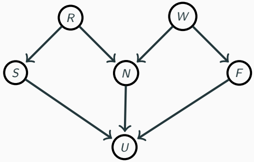


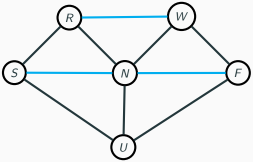
Definition A distribution $P(\mathcal{X}) := P(X_1, ..., X_n)$ factorizes over a *Bayesian Network* $\mathcal{G}$, if P can be expressed as a product:

$$P(X_1, ..., X_n)_{\mathcal{G}} = \prod_{k=1}^{n} P_k(X_k | Pa_{X_k}^{\mathcal{G}}) \overset{\text{as factor}}{=} \prod_{k=1}^{n} \Phi_k(X_k, Pa_{X_k}^{\mathcal{G}}) \tag{5}$$

Definition A distribution $P(\mathcal{X})$ factorizes over a *Markov Network* $\mathcal{H}$, if each $\mathbf{D_k} \subset \mathcal{X}$ $k \in (1, ...K)$ is a complete subgraph (clique) of $\mathcal{H}$:

$$P(X_1, ..., X_n)_{\mathcal{H}} = \frac{1}{Z}\Phi(X_1, ..., X_n) \propto \prod_{k=1}^{K} \Phi_k(\mathbf{D_k}) \tag{6}$$



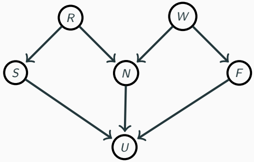


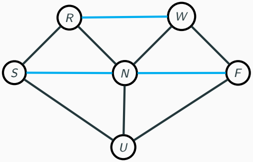
Definition A distribution $P(\mathcal{X}) := P(X_1, ..., X_n)$ factorizes over a *Bayesian Network* $\mathcal{G}$, if P can be expressed as a product:

$$P(X_1, ..., X_n)_{\mathcal{G}} = \prod_{k=1}^{n} P_k(X_k | Pa_{X_k}^{\mathcal{G}}) \overset{as\ factor}{=} \prod_{k=1}^{n} \Phi_k(X_k, Pa_{X_k}^{\mathcal{G}}) \tag{5}$$

Definition A distribution $P(\mathcal{X})$ factorizes over a *Markov Network* $\mathcal{H}$, if each $\mathbf{D_k} \subset \mathcal{X}$ $k \in (1, ...K)$ is a complete subgraph (clique) of $\mathcal{H}$:

$$P(X_1, ..., X_n)_{\mathcal{H}} = \frac{1}{Z} \Phi(X_1, ..., X_n) \propto \prod_{k=1}^{K} \Phi_k(\mathbf{D_k}) \tag{6}$$





Definition The moral graph $\mathcal{M}[G]$ of $\mathcal{G}$ is an undirected graph containing edges between X and Y if there is a directed edge between them or they are both parents of the same node.

FACTOR - NORMALIZATION

factor

$\Phi(\mathbf{X}) : \mathbf{X} \rightarrow \mathbb{R}^+$

i.g. unnormalized measure

Table 1: $\Phi(A, B)$

entry	value
a_0, b_0	π
a_0, b_1	0.1
a_1, b_0	1
a_1, b_1	1000

FACTOR - NORMALIZATION

factor
$\Phi(\mathbf{X}) : \mathbf{X} \rightarrow \mathbb{R}^+$
i.g. unnormalized measure

distribution
$P(\mathbf{X}) : \mathbf{X} \rightarrow [0, 1]$
normalized measure

Table 1: $\Phi(A, B)$

entry	value
a_0, b_0	π
a_0, b_1	0.1
a_1, b_0	1
a_1, b_1	1000

Table 2: $\frac{1}{Z}\Phi(A, B)$

FACTOR - NORMALIZATION

factor

$\Phi(\mathbf{X}) : \mathbf{X} \to \mathbb{R}^+$

i.g. unnormalized measure

distribution

$P(\mathbf{X}) : \mathbf{X} \to [0, 1]$

normalized measure

Table 1: $\Phi(A, B)$

entry	value
a_0, b_0	π
a_0, b_1	0.1
a_1, b_0	1
a_1, b_1	1000

Table 2: $\frac{1}{Z}\Phi(A, B)$

entry	value
a_0, b_0	$3 \cdot 10^{-3}$
a_0, b_1	$1 \cdot 10^{-4}$
a_1, b_0	$1 \cdot 10^{-3}$
a_1, b_1	0.9957

FACTOR - NORMALIZATION

factor
$\Phi(\mathbf{X}) : \mathbf{X} \to \mathbb{R}^+$
i.g. unnormalized measure

distribution
$P(\mathbf{X}) : \mathbf{X} \to [0, 1]$
normalized measure

Table 1: $\Phi(A, B)$

entry	value
a_0, b_0	π
a_0, b_1	0.1
a_1, b_0	1
a_1, b_1	1000

Table 2: $\frac{1}{Z}\Phi(A, B)$

entry	value
a_0, b_0	$3 \cdot 10^{-3}$
a_0, b_1	$1 \cdot 10^{-4}$
a_1, b_0	$1 \cdot 10^{-3}$
a_1, b_1	0.9957

P is called **Gibbs distribution** and $Z = \sum_{\mathbf{x} \in Val(\mathbf{X})} \Phi(\mathbf{x})$ **partition function**.

FACTOR - REDUCTION

factor
$\Phi(\mathbf{Y}, \mathbf{E})$
i.g. unnormalized

reduced factor
$\Phi(\mathbf{Y}, \mathbf{E} = \mathbf{e})$
i.g. unnormalized

Table 3: $\Phi(A, B)$

entry	value
a_0, b_0	π
a_0, b_1	0.1
a_1, b_0	1
a_1, b_1	1000

Table 4: $\Phi(A, B = b_0)$

FACTOR - REDUCTION

factor
$\Phi(\mathbf{Y}, \mathbf{E})$
i.g. unnormalized

reduced factor
$\Phi(\mathbf{Y}, \mathbf{E} = \mathbf{e})$
i.g. unnormalized

Table 3: $\Phi(A, B)$

entry	value
a_0, b_0	π
a_0, b_1	0.1
a_1, b_0	1
a_1, b_1	1000

Table 4: $\Phi(A, B = b_0)$

entry	value
a_0, b_0	π
a_1, b_0	1

FACTOR - REDUCTION

factor
$\Phi(\mathbf{Y}, \mathbf{E})$
i.g. unnormalized

reduced factor
$\Phi(\mathbf{Y}, \mathbf{E} = \mathbf{e})$
i.g. unnormalized

conditional distribution
$P(\mathbf{Y}|\mathbf{E} = \mathbf{e})$
normalized

Table 3: $\Phi(A, B)$

entry	value
a_0, b_0	π
a_0, b_1	0.1
a_1, b_0	1
a_1, b_1	1000

Table 4: $\Phi(A, B = b_0)$

entry	value
a_0, b_0	π
a_1, b_0	1

Table 5: $\frac{1}{Z}\Phi(A, B = b_0)$

entry	value
a_0, b_0	0.76
a_1, b_0	0.24

FACTOR - MARGINALIZATION

factor
$\Phi(\mathbf{Y}, \mathbf{E})$
i.g. unnormalized

reduced factor
$\Psi(\mathbf{Y}) = \sum_{\mathbf{W}} \Phi(\mathbf{Y}, \mathbf{W})$
i.g. unnormalized

Table 6: $\Phi(A, B)$

entry	value
a_0, b_0	π
a_0, b_1	0.1
a_1, b_0	1
a_1, b_1	1000

Table 7: $\sum_{b \in Val(B)} \Phi(A, B = b)$

FACTOR - MARGINALIZATION

factor
$\Phi(\mathbf{Y}, \mathbf{E})$
i.g. unnormalized

reduced factor
$\Psi(\mathbf{Y}) = \sum_{\mathbf{W}} \Phi(\mathbf{Y}, \mathbf{W})$
i.g. unnormalized

Table 6: $\Phi(A, B)$

entry	value
a_0, b_0	π
a_0, b_1	0.1
a_1, b_0	1
a_1, b_1	1000

Table 7: $\sum_{b \in Val(B)} \Phi(A, B = b)$

entry	value
a_0	$\pi + 0.1$
a_1	$1000 + 1$

FACTOR - MARGINALIZATION

factor
$\Phi(\mathbf{Y}, \mathbf{E})$
i.g. unnormalized

reduced factor
$\Psi(\mathbf{Y}) = \sum_{\mathbf{W}} \Phi(\mathbf{Y}, \mathbf{W})$
i.g. unnormalized

marginal distribution
$P(\mathbf{Y})$
normalized

Table 6: $\Phi(A, B)$

entry	value
a_0, b_0	π
a_0, b_1	0.1
a_1, b_0	1
a_1, b_1	1000

Table 7: $\sum_{b \in Val(B)} \Phi(A, B = b)$

entry	value
a_0	$\pi + 0.1$
a_1	$1000 + 1$

Table 8: $\frac{1}{Z}\Psi(A)$

entry	value
a_0	$3 \cdot 10^{-3}$
a_1	0.997

FACTOR - PRODUCT

$\Phi_1(A, C)$

entry	value
a_0, c_0	1
a_0, c_1	2
a_1, c_0	3
a_1, c_1	4

$\Phi_2(B, C)$

entry	value
b_0, c_0	4
b_0, c_1	3
b_1, c_0	2
b_1, c_1	1

FACTOR - PRODUCT

$\Phi_1(A, C)$

entry	value
a_0, c_0	1
a_0, c_1	2
a_1, c_0	3
a_1, c_1	4

$\Phi_2(B, C)$

entry	value
b_0, c_0	4
b_0, c_1	3
b_1, c_0	2
b_1, c_1	1

factor product $\Psi(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$
$= \Phi_1(\mathbf{X}, \mathbf{Z})\Phi_2(\mathbf{Y}, \mathbf{Z})$

Table 9: $\Psi(A, B, C)$

entry	value
a_0, b_0, c_0	$1 \cdot 4$
a_0, b_0, c_1	$2 \cdot 3$
a_0, b_1, c_0	$1 \cdot 2$
a_0, b_1, c_1	$2 \cdot 1$
a_1, b_0, c_0	$3 \cdot 4$
a_1, b_0, c_1	$4 \cdot 3$
a_1, b_1, c_0	$3 \cdot 2$
a_1, b_1, c_1	$4 \cdot 1$

FACTOR - PRODUCT

$\Phi_1(A, C)$

entry	value
a_0, c_0	1
a_0, c_1	2
a_1, c_0	3
a_1, c_1	4

$\Phi_2(B, C)$

entry	value
b_0, c_0	4
b_0, c_1	3
b_1, c_0	2
b_1, c_1	1

factor product $\Psi(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$
$= \Phi_1(\mathbf{X}, \mathbf{Z})\Phi_2(\mathbf{Y}, \mathbf{Z})$

Table 9: $\Psi(A, B, C)$

entry	value
a_0, b_0, c_0	$1 \cdot 4$
a_0, b_0, c_1	$2 \cdot 3$
a_0, b_1, c_0	$1 \cdot 2$
a_0, b_1, c_1	$2 \cdot 1$
a_1, b_0, c_0	$3 \cdot 4$
a_1, b_0, c_1	$4 \cdot 3$
a_1, b_1, c_0	$3 \cdot 2$
a_1, b_1, c_1	$4 \cdot 1$

joint distribution
$P(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$

Table 10: $P(A, B, C)$

entry	value
a_0, b_0, c_0	0.083
a_0, b_0, c_1	0.125
a_0, b_1, c_0	0.042
a_0, b_1, c_1	0.042
a_1, b_0, c_0	0.250
a_1, b_0, c_1	0.250
a_1, b_1, c_0	0.125
a_1, b_1, c_1	0.083

QUERIES

QUERIES - WHAT DO WE WANT TO KNOW?

given are all factors/distributions to factorize the joint $P(\mathcal{X}) \propto \prod_k \Phi_k(\mathbf{D_k})$

QUERIES - WHAT DO WE WANT TO KNOW?

given are all factors/distributions to factorize the joint $P(\mathcal{X}) \propto \prod_k \Phi_k(\mathbf{D_k})$

conditional probability density (CPD): $P(\mathbf{Y} \mid \mathbf{E} = \mathbf{e})$

$\rightarrow$

QUERIES - WHAT DO WE WANT TO KNOW?

given are all factors/distributions to factorize the joint $P(\mathcal{X}) \propto \prod_k \Phi_k(\mathbf{D_k})$

conditional probability density (CPD): $P(\mathbf{Y} \mid \mathbf{E} = \mathbf{e})$

$\rightarrow \frac{P(\mathbf{Y}, \mathbf{E}=\mathbf{e})}{P(\mathbf{E}=\mathbf{e})}$

QUERIES - WHAT DO WE WANT TO KNOW?

given are all factors/distributions to factorize the joint $P(\mathcal{X}) \propto \prod_k \Phi_k(\mathbf{D_k})$

conditional probability density (CPD): $P(\mathbf{Y} \mid \mathbf{E} = \mathbf{e})$

$\rightarrow \frac{P(\mathbf{Y}, \mathbf{E}=\mathbf{e})}{P(\mathbf{E}=\mathbf{e})} \propto P(\mathbf{Y}, \mathbf{E} = \mathbf{e})$

QUERIES - WHAT DO WE WANT TO KNOW?

given are all factors/distributions to factorize the joint $P(\mathcal{X}) \propto \prod_k \Phi_k(\mathbf{D_k})$

conditional probability density (CPD): $P(\mathbf{Y} \mid \mathbf{E} = \mathbf{e})$

$\rightarrow \frac{P(\mathbf{Y},\mathbf{E}=\mathbf{e})}{P(\mathbf{E}=\mathbf{e})} \propto P(\mathbf{Y}, \mathbf{E} = \mathbf{e}) = \sum_{\mathbf{W}} P(\mathbf{Y}, \mathbf{W}, \mathbf{E} = \mathbf{e})$

QUERIES - WHAT DO WE WANT TO KNOW?

given are all factors/distributions to factorize the joint $P(\mathcal{X}) \propto \prod_k \Phi_k(\mathbf{D_k})$

conditional probability density (CPD): $P(\mathbf{Y} \mid \mathbf{E} = \mathbf{e})$

$\rightarrow \frac{P(\mathbf{Y},\mathbf{E}=\mathbf{e})}{P(\mathbf{E}=\mathbf{e})} \propto P(\mathbf{Y}, \mathbf{E} = \mathbf{e}) = \sum_{\mathbf{W}} P(\mathbf{Y}, \mathbf{W}, \mathbf{E} = \mathbf{e}) \propto \sum_{\mathbf{W}} \prod_k \Phi_k(\mathbf{D_k})[\mathbf{E} = \mathbf{e}]$

QUERIES - WHAT DO WE WANT TO KNOW?

given are all factors/distributions to factorize the joint $P(\mathcal{X}) \propto \prod_k \Phi_k(\mathbf{D_k})$

conditional probability density (CPD): $P(\mathbf{Y} \mid \mathbf{E} = \mathbf{e})$

$\rightarrow \frac{P(\mathbf{Y},\mathbf{E}=\mathbf{e})}{P(\mathbf{E}=\mathbf{e})} \propto P(\mathbf{Y}, \mathbf{E} = \mathbf{e}) = \sum_{\mathbf{W}} P(\mathbf{Y}, \mathbf{W}, \mathbf{E} = \mathbf{e}) \propto \sum_{\mathbf{W}} \prod_k \Phi_k(\mathbf{D_k})[\mathbf{E} = \mathbf{e}]$

steps: (a) **reduce** (b) **product** (c) **sum out** (d) **normalize**

QUERIES - WHAT DO WE WANT TO KNOW?

given are all factors/distributions to factorize the joint $P(\mathcal{X}) \propto \prod_k \Phi_k(\mathbf{D_k})$

conditional probability density (CPD): $P(\mathbf{Y} \mid \mathbf{E} = \mathbf{e})$

$\rightarrow \frac{P(\mathbf{Y},\mathbf{E}=\mathbf{e})}{P(\mathbf{E}=\mathbf{e})} \propto P(\mathbf{Y}, \mathbf{E} = \mathbf{e}) = \sum_{\mathbf{W}} P(\mathbf{Y}, \mathbf{W}, \mathbf{E} = \mathbf{e}) \propto \sum_{\mathbf{W}} \prod_k \Phi_k(\mathbf{D_k})[\mathbf{E} = \mathbf{e}]$

steps: (a) **reduce** (b) **product** (c) **sum out** (d) **normalize**

maximum a posteriori (MAP): $\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}}\, P(\mathbf{Y} = \mathbf{y} | \mathbf{E} = \mathbf{e})$

QUERIES - WHAT DO WE WANT TO KNOW?

given are all factors/distributions to factorize the joint $P(\mathcal{X}) \propto \prod_k \Phi_k(\mathbf{D_k})$

conditional probability density (CPD): $P(\mathbf{Y} \mid \mathbf{E} = \mathbf{e})$

$\rightarrow \frac{P(\mathbf{Y},\mathbf{E}=\mathbf{e})}{P(\mathbf{E}=\mathbf{e})} \propto P(\mathbf{Y}, \mathbf{E} = \mathbf{e}) = \sum_{\mathbf{W}} P(\mathbf{Y}, \mathbf{W}, \mathbf{E} = \mathbf{e}) \propto \sum_{\mathbf{W}} \prod_k \Phi_k(\mathbf{D_k})[\mathbf{E} = \mathbf{e}]$

steps: (a) **reduce** (b) **product** (c) **sum out** (d) **normalize**

maximum a posteriori (MAP): $\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} P(\mathbf{Y} = \mathbf{y} | \mathbf{E} = \mathbf{e})$

$\rightarrow$

QUERIES - WHAT DO WE WANT TO KNOW?

given are all factors/distributions to factorize the joint $P(\mathcal{X}) \propto \prod_k \Phi_k(\mathbf{D_k})$

conditional probability density (CPD): $P(\mathbf{Y} \mid \mathbf{E} = \mathbf{e})$

$\rightarrow \frac{P(\mathbf{Y},\mathbf{E}=\mathbf{e})}{P(\mathbf{E}=\mathbf{e})} \propto P(\mathbf{Y}, \mathbf{E} = \mathbf{e}) = \sum_{\mathbf{W}} P(\mathbf{Y}, \mathbf{W}, \mathbf{E} = \mathbf{e}) \propto \sum_{\mathbf{W}} \prod_k \Phi_k(\mathbf{D_k})[\mathbf{E} = \mathbf{e}]$

steps: (a) **reduce** (b) **product** (c) **sum out** (d) **normalize**

maximum a posteriori (MAP): $\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}}\ P(\mathbf{Y} = \mathbf{y} | \mathbf{E} = \mathbf{e})$

$\rightarrow = \underset{y_1,\ldots,y_p}{\operatorname{argmax}}\ log(\prod_k \Phi_k(\mathbf{D_k})[\mathbf{E} = \mathbf{e}])$

QUERIES - WHAT DO WE WANT TO KNOW?

given are all factors/distributions to factorize the joint $P(\mathcal{X}) \propto \prod_k \Phi_k(\mathbf{D_k})$

conditional probability density (CPD): $P(\mathbf{Y} \mid \mathbf{E} = \mathbf{e})$

$\rightarrow \frac{P(\mathbf{Y}, \mathbf{E}=\mathbf{e})}{P(\mathbf{E}=\mathbf{e})} \propto P(\mathbf{Y}, \mathbf{E} = \mathbf{e}) = \sum_{\mathbf{W}} P(\mathbf{Y}, \mathbf{W}, \mathbf{E} = \mathbf{e}) \propto \sum_{\mathbf{W}} \prod_k \Phi_k(\mathbf{D_k})[\mathbf{E} = \mathbf{e}]$

steps: (a) **reduce** (b) **product** (c) **sum out** (d) **normalize**

maximum a posteriori (MAP): $\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}}\ P(\mathbf{Y} = \mathbf{y} | \mathbf{E} = \mathbf{e})$

$\rightarrow = \underset{y_1, \ldots, y_p}{\operatorname{argmax}}\ log(\prod_k \Phi_k(\mathbf{D_k})[\mathbf{E} = \mathbf{e}]) = \underset{y_1, \ldots, y_p}{\operatorname{argmax}}\ \sum_k log(\Phi_k(\mathbf{D_k})[\mathbf{E} = \mathbf{e}])$

QUERIES - WHAT DO WE WANT TO KNOW?

given are all factors/distributions to factorize the joint $P(\mathcal{X}) \propto \prod_k \Phi_k(\mathbf{D_k})$

conditional probability density (CPD): $P(\mathbf{Y} \mid \mathbf{E} = \mathbf{e})$

$\rightarrow \frac{P(\mathbf{Y},\mathbf{E}=\mathbf{e})}{P(\mathbf{E}=\mathbf{e})} \propto P(\mathbf{Y}, \mathbf{E} = \mathbf{e}) = \sum_{\mathbf{W}} P(\mathbf{Y}, \mathbf{W}, \mathbf{E} = \mathbf{e}) \propto \sum_{\mathbf{W}} \prod_k \Phi_k(\mathbf{D_k})[\mathbf{E} = \mathbf{e}]$

steps: (a) **reduce** (b) **product** (c) **sum out** (d) **normalize**

maximum a posteriori (MAP): $\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}}\ P(\mathbf{Y} = \mathbf{y} | \mathbf{E} = \mathbf{e})$

$\rightarrow = \underset{y_1,\dots,y_p}{\operatorname{argmax}}\ log(\prod_k \Phi_k(\mathbf{D_k})[\mathbf{E} = \mathbf{e}]) = \underset{y_1,\dots,y_p}{\operatorname{argmax}}\ \sum_k log(\Phi_k(\mathbf{D_k})[\mathbf{E} = \mathbf{e}])$

steps: (a) **reduce** (b) **search/optimize**

COMPLEXITY

COMPLEXITY - WHY NOT 'JUST' CALCULATE?

- **100 binary** variables $X_1, ..., X_{100}$
- number of entries/values of the joint distribution:

COMPLEXITY - WHY NOT 'JUST' CALCULATE?

- **100 binary** variables $X_1, ..., X_{100}$
- number of entries/values of the joint distribution: 2^{100}

COMPLEXITY - WHY NOT 'JUST' CALCULATE?

- **100 binary** variables $X_1, ..., X_{100}$
- number of entries/values of the joint distribution: 2^{100}
- storage of one entry approximately **1 byte**
- $2^{10} = 1024 \sim 10^3$

COMPLEXITY - WHY NOT 'JUST' CALCULATE?

- **100 binary** variables $X_1, ..., X_{100}$
- number of entries/values of the joint distribution: 2^{100}
- storage of one entry approximately **1 byte**
- $2^{10} = 1024 \sim 10^3 \rightarrow 10^{30}$ bytes or 10^{12} exabytes

COMPLEXITY - WHY NOT 'JUST' CALCULATE?

- **100 binary** variables $X_1, ..., X_{100}$
- number of entries/values of the joint distribution: 2^{100}
- storage of one entry approximately **1 byte**
- $2^{10} = 1024 \sim 10^3 \rightarrow 10^{30}$ bytes or 10^{12} exabytes
- estimated storage of all google servers in 2015 were 10 exabytes

COMPLEXITY - WHY NOT 'JUST' CALCULATE?

- **100 binary** variables $X_1, ..., X_{100}$
- number of entries/values of the joint distribution: 2^{100}
- storage of one entry approximately **1 byte**
- $2^{10} = 1024 \sim 10^3 \rightarrow 10^{30}$ bytes or 10^{12} exabytes
- estimated storage of all google servers in 2015 were 10 exabytes

100 billion times the storage of all google servers

COMPLEXITY - WHY NOT 'JUST' CALCULATE?

- **100 binary** variables $X_1, ..., X_{100}$
- number of entries/values of the joint distribution: 2^{100}
- storage of one entry approximately **1 byte**
- $2^{10} = 1024 \sim 10^3 \rightarrow 10^{30}$ bytes or 10^{12} exabytes
- estimated storage of all google servers in 2015 were 10 exabytes

100 billion times the storage of all google servers, just to store all entries.

COMPLEXITY - CONCLUSION

We can **almost never** calculate:

- the joint distribution factor product!

COMPLEXITY - CONCLUSION

We can **almost never** calculate:

- the joint distribution factor product!
- factor products over a bigger number of variables!

COMPLEXITY - CONCLUSION

We can **almost never** calculate:

- the joint distribution factor product!
- factor products over a bigger number of variables!

COMPLEXITY - CONCLUSION

We can **almost never** calculate:

- the joint distribution factor product!
- factor products over a bigger number of variables!

keep it local

What does effectively reduce variables per factor?

COMPLEXITY - CONCLUSION

We can **almost never** calculate:

- the joint distribution factor product!
- factor products over a bigger number of variables!

keep it local

What does effectively reduce variables per factor?

conditional independence

COMPLEXITY - CONCLUSION

We can **almost never** calculate:

- the joint distribution factor product!
- factor products over a bigger number of variables!

keep it local

What does effectively reduce variables per factor?

conditional independence

clever algorithms

VARIABLE ELIMINATION

VARIABLE ELIMINATION - SCHWAFERTS' GRAPH

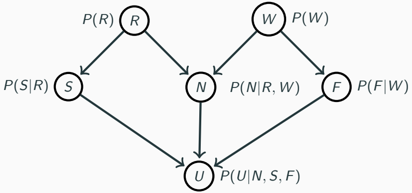


Figure 1: Schwaferts' Graph

VARIABLE ELIMINATION - SCHWAFERTS' GRAPH

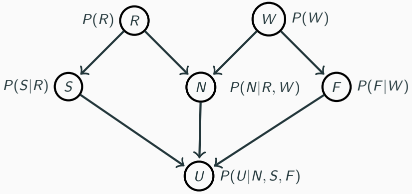


Figure 1: Schwaferts' Graph

query

$$P(U|F=f_1) \overset{convenience}{=} P(U|f_1)$$

VARIABLE ELIMINATION - SCHWAFERTS' GRAPH

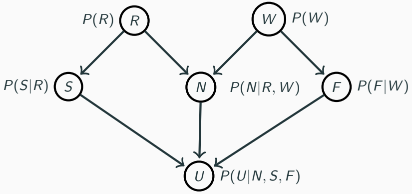


Figure 1: Schwaferts' Graph

query
$P(U|F = f_1) \overset{convenience}{=} P(U|f_1)$

$$P(U|f_1) \propto \sum_{R,W,S,N} P(R)P(W)P(S|R)P(N|R,W)P(f_1|W)P(U|S,N,f_1)$$

VARIABLE ELIMINATION - SCHWAFERTS' GRAPH

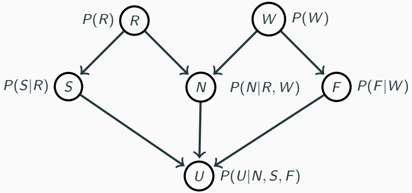


Figure 1: Schwaferts' Graph

query

$P(U|F=f_1) \overset{convenience}{=} P(U|f_1)$

$$P(U|f_1) \propto \sum_{R,W,S,N} P(R)P(W)P(S|R)P(N|R,W)P(f_1|W)P(U|S,N,f_1)$$
$$\propto \sum_{R,S,N} P(R)P(S|R)P(U|S,N,f_1) \sum_{W} P(W)P(N|R,W)P(f_1|W)$$

$$P(U|f_1) \propto \sum_{R,S,N} P(R)P(S|R)P(U|S,N,f_1)\sum_{W} \underbrace{P(W)P(N|R,W)P(f_1|W)}$$

$$P(U|f_1) \propto \sum_{R,S,N} P(R)P(S|R)P(U|S,N,f_1)\sum_{W} \underbrace{P(W)P(N|R,W)P(f_1|W)}$$

product W: $\Psi_1(R,W,N) = P(W)P(N|R,W)P(f_1|W)$

$$P(U|f_1) \propto \sum_{R,S,N} P(R)P(S|R)P(U|S,N,f_1) \sum_W \underbrace{P(W)P(N|R,W)P(f_1|W)}$$

product W: $\Psi_1(R,W,N) = P(W)P(N|R,W)P(f_1|W)$
sum out W: $\tau_1(R,N) = \sum_W \Psi_1(R,W,N)$

$$P(U|f_1) \propto \sum_{R,S,N} P(R)P(S|R)P(U|S,N,f_1) \sum_{W} \underbrace{P(W)P(N|R,W)P(f_1|W)}$$

product W: $\Psi_1(R,W,N) = P(W)P(N|R,W)P(f_1|W)$
sum out W: $\tau_1(R,N) = \sum_W \Psi_1(R,W,N)$

Now there are 3 options to continue:

$$P(U|f_1) \propto \sum_{R,S,N} P(R)P(S|R)P(U|S,N,f_1)\sum_{W} \underbrace{P(W)P(N|R,W)P(f_1|W)}$$

product W: $\Psi_1(R,W,N) = P(W)P(N|R,W)P(f_1|W)$
sum out W: $\tau_1(R,N) = \sum_W \Psi_1(R,W,N)$

Now there are 3 options to continue:
product R: $\Psi_{2R}(R,S,N) = P(R)P(S|R)\tau_1(R,N)$

$$P(U|f_1) \propto \sum_{R,S,N} P(R)P(S|R)P(U|S,N,f_1)\sum_{W} \underbrace{P(W)P(N|R,W)P(f_1|W)}$$

product W: $\Psi_1(R,W,N) = P(W)P(N|R,W)P(f_1|W)$
sum out W: $\tau_1(R,N) = \sum_W \Psi_1(R,W,N)$

Now there are 3 options to continue:
product R: $\Psi_{2R}(R,S,N) = P(R)P(S|R)\tau_1(R,N)$ **mltp:** 2

$$P(U|f_1) \propto \sum_{R,S,N} P(R)P(S|R)P(U|S,N,f_1)\sum_W \underbrace{P(W)P(N|R,W)P(f_1|W)}$$

product W: $\Psi_1(R,W,N) = P(W)P(N|R,W)P(f_1|W)$
sum out W: $\tau_1(R,N) = \sum_W \Psi_1(R,W,N)$

Now there are 3 options to continue:
product R: $\Psi_{2R}(R,S,N) = P(R)P(S|R)\tau_1(R,N)$ **mltp:** $2d_R d_S d_N$

$$P(U|f_1) \propto \sum_{R,S,N} P(R)P(S|R)P(U|S,N,f_1)\sum_W \underbrace{P(W)P(N|R,W)P(f_1|W)}$$

product W: $\Psi_1(R,W,N) = P(W)P(N|R,W)P(f_1|W)$
sum out W: $\tau_1(R,N) = \sum_W \Psi_1(R,W,N)$

Now there are 3 options to continue:
product R: $\Psi_{2R}(R,S,N) = P(R)P(S|R)\tau_1(R,N)$ **mltp:** $2d_R d_S d_N$
product S: $\Psi_{2S}(R,S,N,U) = P(S|R)P(U|S,N,f_1)$ **mltp:** $1\ d_R d_S d_N d_U$

$$P(U|f_1) \propto \sum_{R,S,N} P(R)P(S|R)P(U|S,N,f_1)\sum_W \underbrace{P(W)P(N|R,W)P(f_1|W)}$$

product W: $\Psi_1(R,W,N) = P(W)P(N|R,W)P(f_1|W)$
sum out W: $\tau_1(R,N) = \sum_W \Psi_1(R,W,N)$

Now there are 3 options to continue:
product R: $\Psi_{2R}(R,S,N) = P(R)P(S|R)\tau_1(R,N)$ **mltp:** $2d_R d_S d_N$
product S: $\Psi_{2S}(R,S,N,U) = P(S|R)P(U|S,N,f_1)$ **mltp:** $1\ d_R d_S d_N d_U$
product N: $\Psi_{2N}(R,S,N,U) = \tau_1(R,N)P(U|S,N,f_1)$ **mltp:** $1\ d_R d_S d_N d_U$

$$P(U|f_1) \propto \sum_{R,S,N} P(R)P(S|R)P(U|S,N,f_1)\sum_W \underbrace{P(W)P(N|R,W)P(f_1|W)}$$

product W: $\Psi_1(R,W,N) = P(W)P(N|R,W)P(f_1|W)$
sum out W: $\tau_1(R,N) = \sum_W \Psi_1(R,W,N)$

Now there are 3 options to continue:
product R: $\Psi_{2R}(R,S,N) = P(R)P(S|R)\tau_1(R,N)$ **mltp:** $2d_R d_S d_N$
product S: $\Psi_{2S}(R,S,N,U) = P(S|R)P(U|S,N,f_1)$ **mltp:** $1\ d_R d_S d_N d_U$
product N: $\Psi_{2N}(R,S,N,U) = \tau_1(R,N)P(U|S,N,f_1)$ **mltp:** $1\ d_R d_S d_N d_U$
if U over binary $d_U > 2 \rightarrow$ factor product $2R$ fewest multiplications

$$P(U|f_1) \propto \sum_{R,S,N} P(R)P(S|R)P(U|S,N,f_1)\sum_{W} \underbrace{P(W)P(N|R,W)P(f_1|W)}$$

product W: $\Psi_1(R,W,N) = P(W)P(N|R,W)P(f_1|W)$
sum out W: $\tau_1(R,N) = \sum_W \Psi_1(R,W,N)$

Now there are 3 options to continue:
product R: $\Psi_{2R}(R,S,N) = P(R)P(S|R)\tau_1(R,N)$ **mltp:** $2d_R d_S d_N$
product S: $\Psi_{2S}(R,S,N,U) = P(S|R)P(U|S,N,f_1)$ **mltp:** $1\ d_R d_S d_N d_U$
product N: $\Psi_{2N}(R,S,N,U) = \tau_1(R,N)P(U|S,N,f_1)$ **mltp:** $1\ d_R d_S d_N d_U$
if U over binary $d_U > 2 \rightarrow$ factor product $2R$ fewest multiplications
sum out R: $\tau_2(S,N) = \sum_R \Psi_2(R,S,N)$

$$P(U|f_1) \propto \sum_{R,S,N} P(R)P(S|R)P(U|S,N,f_1)\sum_W \underbrace{P(W)P(N|R,W)P(f_1|W)}$$

product W: $\Psi_1(R,W,N) = P(W)P(N|R,W)P(f_1|W)$
sum out W: $\tau_1(R,N) = \sum_W \Psi_1(R,W,N)$

Now there are 3 options to continue:
product R: $\Psi_{2R}(R,S,N) = P(R)P(S|R)\tau_1(R,N)$ **mltp:** $2d_R d_S d_N$
product S: $\Psi_{2S}(R,S,N,U) = P(S|R)P(U|S,N,f_1)$ **mltp:** $1\ d_R d_S d_N d_U$
product N: $\Psi_{2N}(R,S,N,U) = \tau_1(R,N)P(U|S,N,f_1)$ **mltp:** $1\ d_R d_S d_N d_U$
if U over binary $d_U > 2 \rightarrow$ factor product $2R$ fewest multiplications
sum out R: $\tau_2(S,N) = \sum_R \Psi_2(R,S,N)$

product S,N: $\Psi_3(S,N,U) = P(U|S,N,f_1)\tau_2(S,N)$

$$P(U|f_1) \propto \sum_{R,S,N} P(R)P(S|R)P(U|S,N,f_1)\sum_{W} \underbrace{P(W)P(N|R,W)P(f_1|W)}$$

product W: $\Psi_1(R,W,N) = P(W)P(N|R,W)P(f_1|W)$
sum out W: $\tau_1(R,N) = \sum_W \Psi_1(R,W,N)$

Now there are 3 options to continue:
product R: $\Psi_{2R}(R,S,N) = P(R)P(S|R)\tau_1(R,N)$ **mltp:** $2d_R d_S d_N$
product S: $\Psi_{2S}(R,S,N,U) = P(S|R)P(U|S,N,f_1)$ **mltp:** $1\ d_R d_S d_N d_U$
product N: $\Psi_{2N}(R,S,N,U) = \tau_1(R,N)P(U|S,N,f_1)$ **mltp:** $1\ d_R d_S d_N d_U$
if U over binary $d_U > 2 \rightarrow$ factor product $2R$ fewest multiplications
sum out R: $\tau_2(S,N) = \sum_R \Psi_2(R,S,N)$

product S,N: $\Psi_3(S,N,U) = P(U|S,N,f_1)\tau_2(S,N)$
sum out S,N: $\sum_{S,N} \Psi_3(S,N,U)$

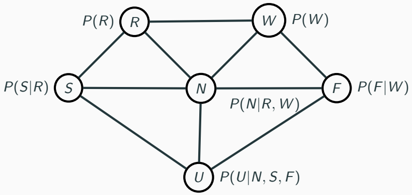
$$P(U|f_1) \propto \sum_{R,S,N} P(R)P(S|R)P(U|S,N,f_1)\sum_{W} \underbrace{P(W)P(N|R,W)P(f_1|W)}$$

product W: $\Psi_1(R,W,N) = P(W)P(N|R,W)P(f_1|W)$
sum out W: $\tau_1(R,N) = \sum_W \Psi_1(R,W,N)$

Now there are 3 options to continue:
product R: $\Psi_{2R}(R,S,N) = P(R)P(S|R)\tau_1(R,N)$ **mltp:** $2d_R d_S d_N$
product S: $\Psi_{2S}(R,S,N,U) = P(S|R)P(U|S,N,f_1)$ **mltp:** $1\ d_R d_S d_N d_U$
product N: $\Psi_{2N}(R,S,N,U) = \tau_1(R,N)P(U|S,N,f_1)$ **mltp:** $1\ d_R d_S d_N d_U$
if U over binary $d_U > 2 \rightarrow$ factor product $2R$ fewest multiplications
sum out R: $\tau_2(S,N) = \sum_R \Psi_2(R,S,N)$

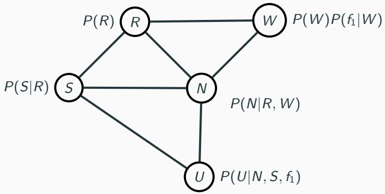
product S,N: $\Psi_3(S,N,U) = P(U|S,N,f_1)\tau_2(S,N)$
sum out S,N: $\sum_{S,N} \Psi_3(S,N,U) \propto P(U|f_1)$

VARIABLE ELIMINATION - GRAPH VIEW



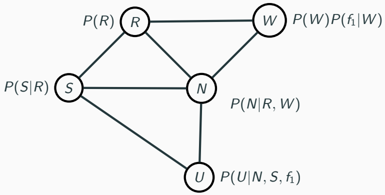
- **reduce:** $F = f_1$

VARIABLE ELIMINATION - GRAPH VIEW



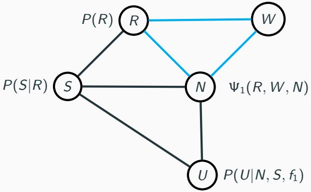
- **reduce:** $F = f_1$

VARIABLE ELIMINATION - GRAPH VIEW



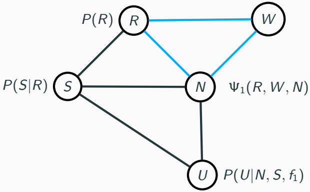
- **reduce:** $F = f_1$
- **product W:** $\Psi_1(R, W, N)$

VARIABLE ELIMINATION - GRAPH VIEW



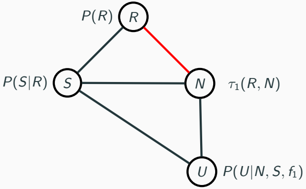
- **reduce:** $F = f_1$
- **product W:** $\Psi_1(R, W, N)$

VARIABLE ELIMINATION - GRAPH VIEW



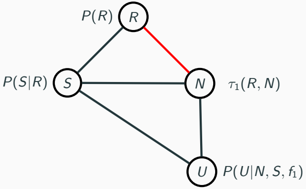
- **reduce:** $F = f_1$
- **product W:** $\Psi_1(R, W, N)$
- **sum out W:** $\tau_1(R, N)$

VARIABLE ELIMINATION - GRAPH VIEW



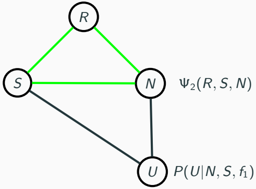
- **reduce:** $F = f_1$
- **product W:** $\Psi_1(R, W, N)$
- **sum out W:** $\tau_1(R, N)$

VARIABLE ELIMINATION - GRAPH VIEW



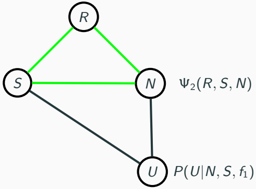
- **reduce:** $F = f_1$
- **product W:** $\Psi_1(R, W, N)$
- **sum out W:** $\tau_1(R, N)$
- **product R:** $\Psi_2(R, S, N)$

VARIABLE ELIMINATION - GRAPH VIEW



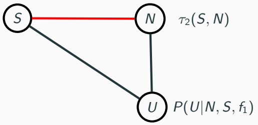
- **reduce:** $F = f_1$
- **product W:** $\Psi_1(R, W, N)$
- **sum out W:** $\tau_1(R, N)$
- **product R:** $\Psi_2(R, S, N)$

VARIABLE ELIMINATION - GRAPH VIEW



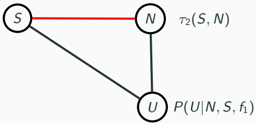
- **reduce:** $F = f_1$
- **product W:** $\Psi_1(R, W, N)$
- **sum out W:** $\tau_1(R, N)$
- **product R:** $\Psi_2(R, S, N)$
- **sum out R:** $\tau_2(S, N)$

VARIABLE ELIMINATION - GRAPH VIEW



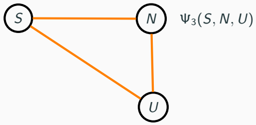
- **reduce:** $F = f_1$
- **product W:** $\Psi_1(R, W, N)$
- **sum out W:** $\tau_1(R, N)$
- **product R:** $\Psi_2(R, S, N)$
- **sum out R:** $\tau_2(S, N)$

VARIABLE ELIMINATION - GRAPH VIEW



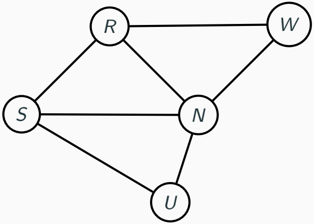
- **reduce:** $F = f_1$
- **product W:** $\Psi_1(R, W, N)$
- **sum out W:** $\tau_1(R, N)$
- **product R:** $\Psi_2(R, S, N)$
- **sum out R:** $\tau_2(S, N)$
- **product R:** $\Psi_3(S, N, U)$

VARIABLE ELIMINATION - VIEW

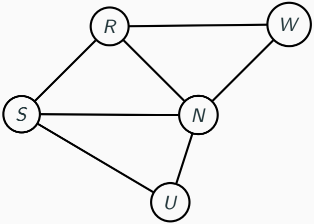


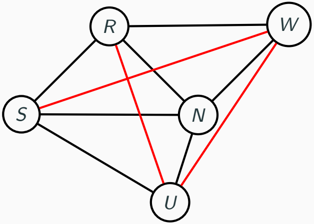
- **reduce:** $F = f_1$
- **product W:** $\Psi_1(R, W, N)$
- **sum out W:** $\tau_1(R, N)$
- **product R:** $\Psi_2(R, S, N)$
- **sum out R:** $\tau_2(S, N)$
- **product R:** $\Psi_3(S, N, U)$

VARIABLE ELIMINATION - INDUCED GRAPH

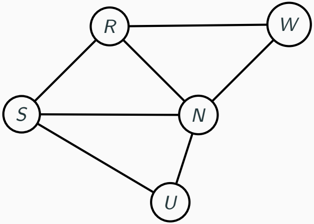


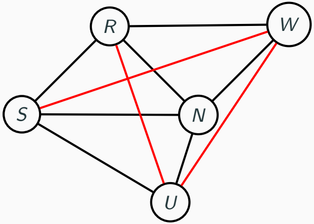
VARIABLE ELIMINATION - INDUCED GRAPH



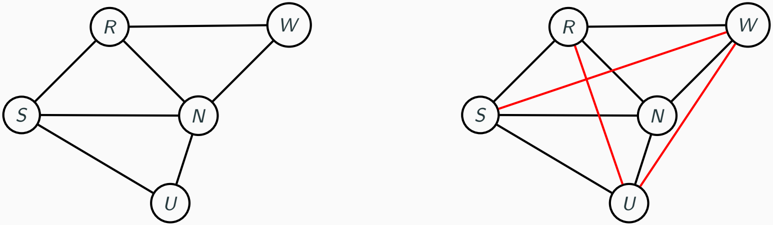


VARIABLE ELIMINATION - INDUCED GRAPH





Let $\oplus = \{\Phi_1, ..., \Phi_K\}$ be a set of factors over $\mathcal{X} = \{X_1, ..., X_n\}$, and α be an elimination ordering for some subset $\mathbf{X} \subseteq \mathcal{X}$. The induced graph $\mathcal{I}_{\oplus,\alpha}$ is an undirected graph over $\mathcal{X}$, where X_i and X_j are connected by en edge if they both appear in same intermediate factor Ψ genreated by the VE algorithm using α as an elimination ordering.

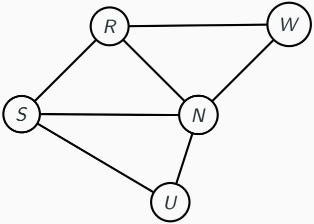


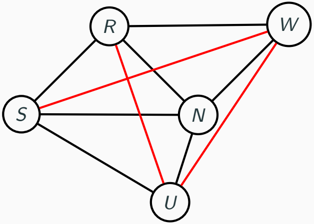
Theorem

The scope of every generated factor Ψ is a clique in $\mathcal{I}_{\oplus,\alpha}$.

Every maximal clique in $\mathcal{I}_{\oplus,\alpha}$ is the scope of some Ψ.

VARIABLE ELIMINATION - INDUCED GRAPH

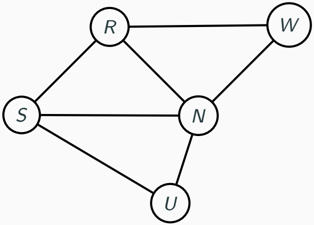


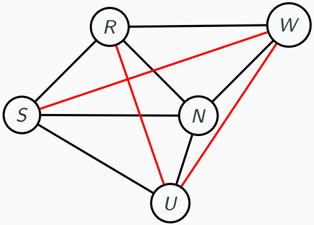


Theorem

The scope of every generated factor Ψ is a clique in $\mathcal{I}_{\oplus,\alpha}$.
Every maximal clique in $\mathcal{I}_{\oplus,\alpha}$ is the scope of some Ψ.

$\rightarrow$ Direct correspondence: maximal factors generated $\leftrightarrow$ maximal cliques.



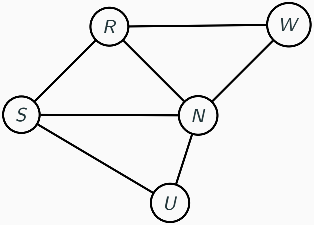


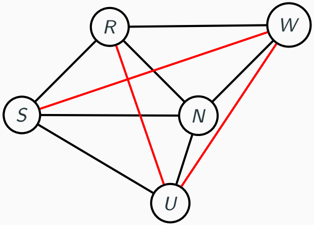
Theorem

The scope of every generated factor Ψ is a clique in $\mathcal{I}_{\oplus,\alpha}$.
Every maximal clique in $\mathcal{I}_{\oplus,\alpha}$ is the scope of some Ψ.

$\rightarrow$ Direct correspondence: maximal factors generated $\leftrightarrow$ maximal cliques.
$\rightarrow$ Size of maximal cliques depends strongly on α.

VARIABLE ELIMINATION - INDUCED GRAPH



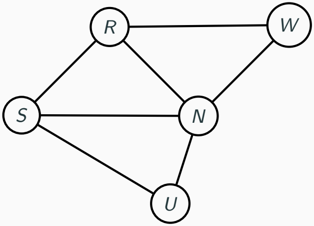


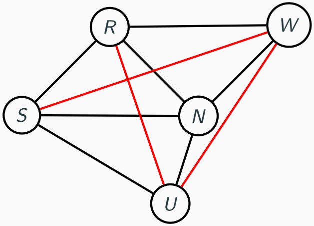
Theorem

The scope of every generated factor Ψ is a clique in $\mathcal{I}_{\oplus,\alpha}$.
Every maximal clique in $\mathcal{I}_{\oplus,\alpha}$ is the scope of some Ψ.

$\rightarrow$ Direct correspondence: maximal factors generated $\leftrightarrow$ maximal cliques.
$\rightarrow$ Size of maximal cliques depends strongly on α.
width: Number of variables in largest clique in $\mathcal{I}_{\oplus,\alpha}$ minus 1.

VARIABLE ELIMINATION - INDUCED GRAPH





Theorem

The scope of every generated factor Ψ is a clique in $\mathcal{I}_{\oplus,\alpha}$.
Every maximal clique in $\mathcal{I}_{\oplus,\alpha}$ is the scope of some Ψ.

$\rightarrow$ Direct correspondence: maximal factors generated $\leftrightarrow$ maximal cliques.
$\rightarrow$ Size of maximal cliques depends strongly on α.
width: Number of variables in largest clique in $\mathcal{I}_{\oplus,\alpha}$ minus 1.

Finding best α is *NP-complete*, but search algorithms using simple heuristics (min-neighbor/weight/fill) work in practice.

VARIABLE ELIMINATION - SUMMARY

Algorithm 1 Variable Elimination for CPD

1: *reduce* initial factors by evidence $\mathbf{E} = \mathbf{e} \rightarrow \oplus := \{\Phi_1, ..., \Phi_m\}$

VARIABLE ELIMINATION - SUMMARY

Algorithm 2 Variable Elimination for CPD

1: *reduce* initial factors by evidence $\mathbf{E} = \mathbf{e} \rightarrow \oplus := \{\Phi_1, ..., \Phi_m\}$

2: *search* elimination ordering using heuristics $\rightarrow \alpha$

VARIABLE ELIMINATION - SUMMARY

Algorithm 3 Variable Elimination for CPD

1: *reduce* initial factors by evidence $\mathbf{E} = \mathbf{e} \rightarrow \oplus := \{\Phi_1, ..., \Phi_m\}$

2: *search* elimination ordering using heuristics $\rightarrow \alpha$

3: **for all** $X_{\alpha(k)} \in \mathbf{W}$ **do**

VARIABLE ELIMINATION - SUMMARY

Algorithm 4 Variable Elimination for CPD

reduce initial factors by evidence $\mathbf{E} = \mathbf{e} \rightarrow \oplus := \{\Phi_1, ..., \Phi_m\}$

search elimination ordering using heuristics $\rightarrow \alpha$

for all $X_{\alpha(k)} \in \mathbf{W}$ **do**

    product $\Psi_k = \prod_{\Phi_i \in \oplus : X_{\alpha(k)} \in Scope[\Phi_i]} \Phi_i$

VARIABLE ELIMINATION - SUMMARY

Algorithm 5 Variable Elimination for CPD

1: *reduce* initial factors by evidence $\mathbf{E} = \mathbf{e} \rightarrow \oplus := \{\Phi_1, ..., \Phi_m\}$

2: *search* elimination ordering using heuristics $\rightarrow \alpha$

3: **for all** $X_{\alpha(k)} \in \mathbf{W}$ **do**

4: *product* $\Psi_k = \prod_{\Phi_i \in \oplus : X_{\alpha(k)} \in Scope[\Phi_i]} \Phi_i$

5: *sum* $\tau_k = \sum_{X_{\alpha(k)}} \Psi_k$

VARIABLE ELIMINATION - SUMMARY

Algorithm 6 Variable Elimination for CPD

reduce initial factors by evidence $\mathbf{E} = \mathbf{e} \rightarrow \oplus := \{\Phi_1, ..., \Phi_m\}$
search elimination ordering using heuristics $\rightarrow \alpha$
for all $X_{\alpha(k)} \in \mathbf{W}$ **do**
    product $\Psi_k = \prod_{\Phi_i \in \oplus : X_{\alpha(k)} \in Scope[\Phi_i]} \Phi_i$
    sum $\tau_k = \sum_{X_{\alpha(k)}} \Psi_k$
    update $\tau_k := \Phi_{m+k} \in \oplus$

VARIABLE ELIMINATION - SUMMARY

Algorithm 7 Variable Elimination for CPD

reduce initial factors by evidence $\mathbf{E} = \mathbf{e} \rightarrow \oplus := \{\Phi_1, ..., \Phi_m\}$

search elimination ordering using heuristics $\rightarrow \alpha$

for all $X_{\alpha(k)} \in \mathbf{W}$ **do**

  product $\Psi_k = \prod_{\Phi_i \in \oplus : X_{\alpha(k)} \in Scope[\Phi_i]} \Phi_i$

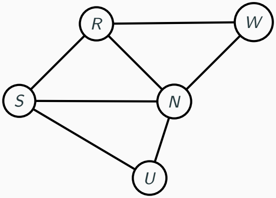
  sum $\tau_k = \sum_{X_{\alpha(k)}} \Psi_k$

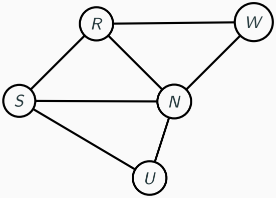
  update $\tau_k := \Phi_{m+k} \in \oplus$

end for

renormalize the final product $\Psi_{K+1}(\mathbf{Y}) \rightarrow$ proper CDP $P(\mathbf{Y}|\mathbf{E} = \mathbf{e})$

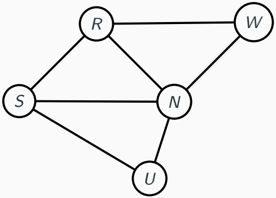
MESSAGE PASSING


R
W
S
N
U



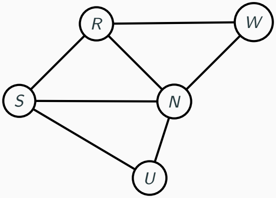
factors generated by VE

- $\Psi_1(R, W, N)$ cluster
- $\tau_1(R, N)$ sepset
- $\Psi_2(R, S, N)$ cluster
- $\tau_2(S, N)$ sepset
- $\Psi_3(S, N, U)$ cluster



factors generated by VE

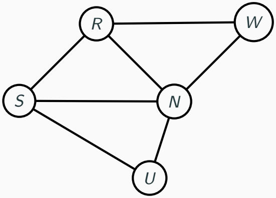
- $\Psi_1(R, W, N)$ cluster
- $\tau_1(R, N)$ sepset
- $\Psi_2(R, S, N)$ cluster
- $\tau_2(S, N)$ sepset
- $\Psi_3(S, N, U)$ cluster



factors generated by VE

- $\Psi_1(R, W, N)$ cluster
- $\tau_1(R, N)$ sepset
- $\Psi_2(R, S, N)$ cluster
- $\tau_2(S, N)$ sepset
- $\Psi_3(S, N, U)$ cluster

A **cluster graph** $\mathcal{T}$ is an undirected graph.
Each node is associated with a cluster $\mathbf{C_i} \subseteq \mathcal{X}$.
Each edge is associated with a sepset $\mathbf{S_{i,j}} \supseteq \mathbf{C_i} \cap \mathbf{C_j}$.
Family-preserving: Each factor $\Phi_k \in \oplus$ must be associated with a cluster $\mathbf{C_k}$, such that $Scope[\Phi_k] \supseteq \mathbf{C_k}$.

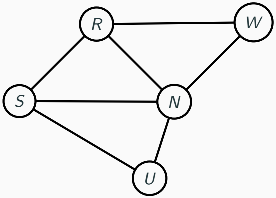


factors generated by VE

- $\Psi_1(R, W, N)$ cluster
- $\tau_1(R, N)$ sepset
- $\Psi_2(R, S, N)$ cluster
- $\tau_2(S, N)$ sepset
- $\Psi_3(S, N, U)$ cluster

A **cluster graph** $\mathcal{T}$ is an undirected graph.
Each node is associated with a cluster $\mathbf{C_i} \subseteq \mathcal{X}$.
Each edge is associated with a sepset $\mathbf{S_{i,j}} \supseteq \mathbf{C_i} \cap \mathbf{C_j}$.
Family-preserving: Each factor $\Phi_k \in \oplus$ must be associated with a cluster $\mathbf{C_k}$, such that $Scope[\Phi_k] \supseteq \mathbf{C_k}$.

Running Intersection Property: All information has to be shared on one way.

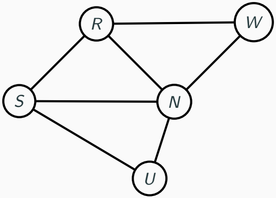


factors generated by VE

- $\Psi_1(R, W, N)$ cluster
- $\tau_1(R, N)$ sepset
- $\Psi_2(R, S, N)$ cluster
- $\tau_2(S, N)$ sepset
- $\Psi_3(S, N, U)$ cluster

A **cluster graph** $\mathcal{T}$ is an undirected graph.
Each node is associated with a cluster $\mathbf{C_i} \subseteq \mathcal{X}$.
Each edge is associated with a sepset $\mathbf{S_{i,j}} \supseteq \mathbf{C_i} \cap \mathbf{C_j}$.
Family-preserving: Each factor $\Phi_k \in \oplus$ must be associated with a cluster $\mathbf{C_k}$, such that $Scope[\Phi_k] \supseteq \mathbf{C_k}$.

Running Intersection Property: All information has to be shared on one way.
Clique Tree: Cluster graph without loops that satisfies the *RIP*.

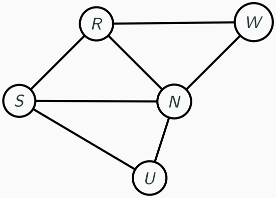


factors generated by VE

- $\Psi_1(R, W, N)$ cluster
- $\tau_1(R, N)$ sepset
- $\Psi_2(R, S, N)$ cluster
- $\tau_2(S, N)$ sepset
- $\Psi_3(S, N, U)$ cluster

A **cluster graph** $\mathcal{T}$ is an undirected graph.
Each node is associated with a cluster $\mathbf{C_i} \subseteq \mathcal{X}$.
Each edge is associated with a sepset $\mathbf{S_{i,j}} \supseteq \mathbf{C_i} \cap \mathbf{C_j}$.
Family-preserving: Each factor $\Phi_k \in \oplus$ must be associated with a cluster $\mathbf{C_k}$, such that $Scope[\Phi_k] \supseteq \mathbf{C_k}$.

Running Intersection Property: All information has to be shared on one way.
Clique Tree: Cluster graph without loops that satisfies the *RIP*.
Theorem: A Variable Elimination process induces a clique tree.

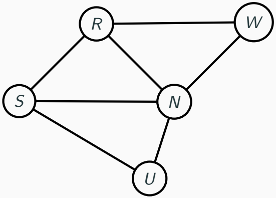


factors generated by VE

- $\Psi_1(R, W, N)$ cluster
- $\tau_1(R, N)$ sepset
- $\Psi_2(R, S, N)$ cluster
- $\tau_2(S, N)$ sepset
- $\Psi_3(S, N, U)$ cluster

A **cluster graph** $\mathcal{T}$ is an undirected graph.
Each node is associated with a cluster $\mathbf{C_i} \subseteq \mathcal{X}$.
Each edge is associated with a sepset $\mathbf{S_{i,j}} \supseteq \mathbf{C_i} \cap \mathbf{C_j}$.
Family-preserving: Each factor $\Phi_k \in \oplus$ must be associated with a cluster $\mathbf{C_k}$, such that $Scope[\Phi_k] \supseteq \mathbf{C_k}$.

Running Intersection Property: All information has to be shared on one way.
Clique Tree: Cluster graph without loops that satisfies the *RIP*.
Theorem: A Variable Elimination process induces a clique tree.
Correctness: Exact marginals for clique trees, approximate for loopy cluster graphs.



factors generated by VE

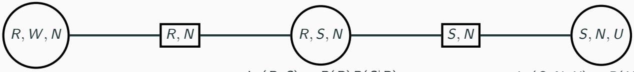
- $\Psi_1(R, W, N)$ cluster
- $\tau_1(R, N)$ sepset
- $\Psi_2(R, S, N)$ cluster
- $\tau_2(S, N)$ sepset
- $\Psi_3(S, N, U)$ cluster

A **cluster graph** $\mathcal{T}$ is an undirected graph.
Each node is associated with a cluster $\mathbf{C_i} \subseteq \mathcal{X}$.
Each edge is associated with a sepset $\mathbf{S_{i,j}} \supseteq \mathbf{C_i} \cap \mathbf{C_j}$.
Family-preserving: Each factor $\Phi_k \in \oplus$ must be associated with a cluster $\mathbf{C_k}$, such that $Scope[\Phi_k] \supseteq \mathbf{C_k}$.

Running Intersection Property: All information has to be shared on one way.
Clique Tree: Cluster graph without loops that satisfies the *RIP*.
Theorem: A Variable Elimination process induces a clique tree.
Correctness: Exact marginals for clique trees, approximate for loopy cluster graphs.

MESSAGE PASSING

initial potential Ψ_k



$\Psi_1(R, W, N) = P(W)P(N|R, W)P(f_1|W)$

$\Psi_2(R, S) = P(R)P(S|R)$

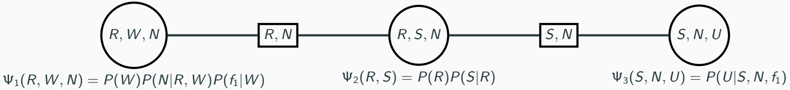
$\Psi_3(S, N, U) = P(U|S, N, f_1)$

MESSAGE PASSING

initial potential: Ψ_k

message from $\mathbf{C_i}$ to $\mathbf{C_j}$: $\delta_{i\to j}(\mathbf{S_{i,j}}) = \sum_{\mathbf{C_i}-\mathbf{S_{i,j}}} \Psi_i \prod_{k\in(\mathcal{N}_i-\{j\})} \delta_{k\to i}$

$$\delta_{1\to 2} = \Sigma_W \, \Psi_1$$



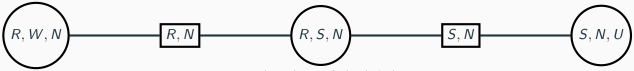
$$\delta_{3\to 2} = \Sigma_U \, \Psi_3$$

MESSAGE PASSING

initial potential: Ψ_k

message from $\mathbf{C_i}$ to $\mathbf{C_j}$: $\delta_{i\to j}(\mathbf{S_{i,j}}) = \sum_{\mathbf{C_i}-\mathbf{S_{i,j}}} \Psi_i \prod_{k\in(\mathcal{N}_i-\{j\})} \delta_{k\to i}$

$\delta_{1\to 2} = \Sigma_W\, \Psi_1$ $\quad$ $\delta_{2\to 3} = \Sigma_R\, \Psi_2 \delta_{1\to 2}$



$\Psi_1(R, W, N) = P(W)P(N|R, W)P(f_1|W)$ $\quad$ $\Psi_2(R, S) = P(R)P(S|R)$ $\quad$ $\Psi_3(S, N, U) = P(U|S, N, f_1)$

$\delta_{2\to 1} = \Sigma_S\, \Psi_2 \delta_{3\to 2}$ $\quad$ $\delta_{3\to 2} = \Sigma_U\, \Psi_3$

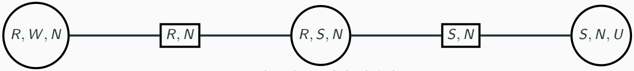
MESSAGE PASSING

initial potential: Ψ_k

message from $\mathbf{C_i}$ to $\mathbf{C_j}$: $\delta_{i\to j}(\mathbf{S_{i,j}}) = \sum_{\mathbf{C_i}-\mathbf{S_{i,j}}} \Psi_i \prod_{k\in(\mathcal{N}_i-\{j\})} \delta_{k\to i}$

$\delta_{1\to 2} = \Sigma_W\, \Psi_1$

$\delta_{2\to 3} = \Sigma_R\, \Psi_2 \delta_{1\to 2}$



$\Psi_1(R, W, N) = P(W)P(N|R, W)P(f_1|W)$

$\Psi_2(R, S) = P(R)P(S|R)$

$\Psi_3(S, N, U) = P(U|S, N, f_1)$

$\delta_{2\to 1} = \Sigma_S\, \Psi_2 \delta_{3\to 2}$

$\delta_{3\to 2} = \Sigma_U\, \Psi_3$

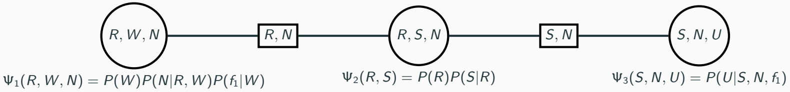
Beliefs: $\propto$ marginals(exact/approximate): $\beta_i(\mathbf{C_i}) = \Psi_i \prod_{k\in\mathcal{N}_i} \delta_{k\to i}$

MESSAGE PASSING

initial potential: Ψ_k

message from $\mathbf{C_i}$ to $\mathbf{C_j}$: $\delta_{i\to j}(\mathbf{S_{i,j}}) = \sum_{\mathbf{C_i}-\mathbf{S_{i,j}}} \Psi_i \prod_{k\in(\mathcal{N}_i-\{j\})} \delta_{k\to i}$

$\delta_{1\to 2} = \Sigma_W \Psi_1$ $\qquad$ $\delta_{2\to 3} = \Sigma_R \Psi_2 \delta_{1\to 2}$



$\delta_{2\to 1} = \Sigma_S \Psi_2 \delta_{3\to 2}$ $\qquad$ $\delta_{3\to 2} = \Sigma_U \Psi_3$

Beliefs: $\propto$ marginals(exact/approximate): $\beta_i(\mathbf{C_i}) = \Psi_i \prod_{k\in\mathcal{N}_i} \delta_{k\to i}$

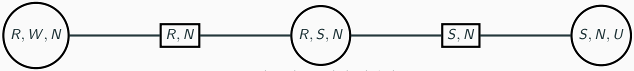
Calibration: clusters agree in their beliefs over their sepset:

$\sum_{\mathbf{C_i}-\mathbf{S_{i,j}}} \beta_i(\mathbf{C_i}) = \sum_{\mathbf{C_j}-\mathbf{S_{i,j}}} \beta_j(\mathbf{C_j})$

initial potential: Ψ_k

message from $\mathbf{C_i}$ to $\mathbf{C_j}$: $\delta_{i\to j}(\mathbf{S_{i,j}}) = \sum_{\mathbf{C_i}-\mathbf{S_{i,j}}} \Psi_i \prod_{k\in(\mathcal{N}_i-\{j\})} \delta_{k\to i}$

$\delta_{1\to 2} = \Sigma_W \Psi_1$ $\delta_{2\to 3} = \Sigma_R \Psi_2 \delta_{1\to 2}$



$\Psi_1(R,W,N) = P(W)P(N|R,W)P(f_1|W)$ $\Psi_2(R,S) = P(R)P(S|R)$ $\Psi_3(S,N,U) = P(U|S,N,f_1)$

$\delta_{2\to 1} = \Sigma_S \Psi_2 \delta_{3\to 2}$ $\delta_{3\to 2} = \Sigma_U \Psi_3$

Beliefs: $\propto$ marginals(exact/approximate): $\beta_i(\mathbf{C_i}) = \Psi_i \prod_{k\in\mathcal{N}_i} \delta_{k\to i}$
Calibration: clusters agree in their beliefs over their sepset:
$\sum_{\mathbf{C_i}-\mathbf{S_{i,j}}} \beta_i(\mathbf{C_i}) = \sum_{\mathbf{C_j}-\mathbf{S_{i,j}}} \beta_j(\mathbf{C_j})$
Result: $P(R,W,N)$, $P(R,S,N)$ and $P(S,N,U)$.

How do I get $P(U|S=s_1)$ or $P(U|W=w_1)$?

MESSAGE PASSING - CLIQUE TREE ALGORITHM

Algorithm 8 Clique Tree Algorithm

1: **simulate** VE and search for good $\alpha \rightarrow$ clique tree.

MESSAGE PASSING - CLIQUE TREE ALGORITHM

Algorithm 9 Clique Tree Algorithm

1: **simulate** VE and search for good $\alpha \to$ clique tree.

2: **product** initial potentials: $\Psi_i(\mathbf{C_i}) = \prod_{k:\alpha(k)=i} \Phi_k$.

Algorithm 10 Clique Tree Algorithm

1: **simulate** VE and search for good $\alpha \rightarrow$ clique tree.

2: **product** initial potentials: $\Psi_i(\mathbf{C_i}) = \prod_{k:\alpha(k)=i} \Phi_k$.

3: **update** messages: $\delta_{i\rightarrow j}(\mathbf{S_{i,j}}) = \sum_{\mathbf{C_i}-\mathbf{S_{i,j}}} \Psi_i \prod_{k\in(\mathcal{N}_i-j)} \delta_{k\rightarrow i}$.

MESSAGE PASSING - CLIQUE TREE ALGORITHM

Algorithm 11 Clique Tree Algorithm

1: **simulate** VE and search for good $\alpha \to$ clique tree.
2: **product** initial potentials: $\Psi_i(\mathbf{C_i}) = \prod_{k:\alpha(k)=i} \Phi_k$.
3: **update** messages: $\delta_{i\to j}(\mathbf{S_{i,j}}) = \sum_{\mathbf{C_i}-\mathbf{S_{i,j}}} \Psi_i \prod_{k\in(\mathcal{N}_i-j)} \delta_{k\to i}$.
4: **upward pass** all messages from leafs to root.

MESSAGE PASSING - CLIQUE TREE ALGORITHM

Algorithm 12 Clique Tree Algorithm

1: **simulate** VE and search for good $\alpha \rightarrow$ clique tree.

2: **product** initial potentials: $\Psi_i(\mathbf{C_i}) = \prod_{k:\alpha(k)=i} \Phi_k$.

3: **update** messages: $\delta_{i\rightarrow j}(\mathbf{S_{i,j}}) = \sum_{\mathbf{C_i}-\mathbf{S_{i,j}}} \Psi_i \prod_{k\in(\mathcal{N}_i-j)} \delta_{k\rightarrow i}$.

4: **upward pass** all messages from leafs to root.

5: **downward pass** all messages from root to leafs.

MESSAGE PASSING - CLIQUE TREE ALGORITHM

Algorithm 13 Clique Tree Algorithm

1: **simulate** VE and search for good $\alpha \rightarrow$ clique tree.
2: **product** initial potentials: $\Psi_i(\mathbf{C_i}) = \prod_{k:\alpha(k)=i} \Phi_k$.
3: **update** messages: $\delta_{i\rightarrow j}(\mathbf{S_{i,j}}) = \sum_{\mathbf{C_i}-\mathbf{S_{i,j}}} \Psi_i \prod_{k\in(\mathcal{N}_i-j)} \delta_{k\rightarrow i}$.
4: **upward pass** all messages from leafs to root.
5: **downward pass** all messages from root to leafs.
6: **calculate** beliefs: $\beta_i(\mathbf{C_i}) = \Psi_i \prod_{k\in\mathcal{N}_i} \delta_{k\rightarrow i}$.

MESSAGE PASSING - CLIQUE TREE ALGORITHM

Algorithm 14 Clique Tree Algorithm

1: **simulate** VE and search for good $\alpha \to$ clique tree.
2: **product** initial potentials: $\Psi_i(\mathbf{C_i}) = \prod_{k:\alpha(k)=i} \Phi_k$.
3: **update** messages: $\delta_{i \to j}(\mathbf{S_{i,j}}) = \sum_{\mathbf{C_i} - \mathbf{S_{i,j}}} \Psi_i \prod_{k \in (\mathcal{N}_i - j)} \delta_{k \to i}$.
4: **upward pass** all messages from leafs to root.
5: **downward pass** all messages from root to leafs.
6: **calculate** beliefs: $\beta_i(\mathbf{C_i}) = \Psi_i \prod_{k \in \mathcal{N}_i} \delta_{k \to i}$.
7: **renormalize** beliefs to get marginal distributions.

MESSAGE PASSING - CLIQUE TREE ALGORITHM

Algorithm 15 Clique Tree Algorithm

1: **simulate** VE and search for good $\alpha \rightarrow$ clique tree.
2: **product** initial potentials: $\Psi_i(\mathbf{C_i}) = \prod_{k:\alpha(k)=i} \Phi_k$.
3: **update** messages: $\delta_{i\rightarrow j}(\mathbf{S_{i,j}}) = \sum_{\mathbf{C_i}-\mathbf{S_{i,j}}} \Psi_i \prod_{k\in(\mathcal{N}_i-j)} \delta_{k\rightarrow i}$.
4: **upward pass** all messages from leafs to root.
5: **downward pass** all messages from root to leafs.
6: **calculate** beliefs: $\beta_i(\mathbf{C_i}) = \Psi_i \prod_{k\in\mathcal{N}_i} \delta_{k\rightarrow i}$.
7: **renormalize** beliefs to get marginal distributions.
8: **store** tree structure and marginal distributions.

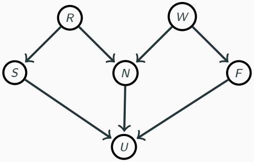
MESSAGE PASSING - CLIQUE TREE ALGORITHM

Algorithm 16 Clique Tree Algorithm

1: **simulate** VE and search for good $\alpha \rightarrow$ clique tree.
2: **product** initial potentials: $\Psi_i(\mathbf{C_i}) = \prod_{k:\alpha(k)=i} \Phi_k$.
3: **update** messages: $\delta_{i\rightarrow j}(\mathbf{S_{i,j}}) = \sum_{\mathbf{C_i}-\mathbf{S_{i,j}}} \Psi_i \prod_{k\in(\mathcal{N}_i-j)} \delta_{k\rightarrow i}$.
4: **upward pass** all messages from leafs to root.
5: **downward pass** all messages from root to leafs.
6: **calculate** beliefs: $\beta_i(\mathbf{C_i}) = \Psi_i \prod_{k\in\mathcal{N}_i} \delta_{k\rightarrow i}$.
7: **renormalize** beliefs to get marginal distributions.
8: **store** tree structure and marginal distributions.

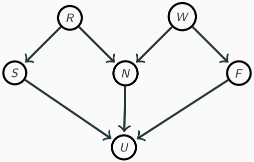
SAMPLING

SAMPLING



simple forward sampling

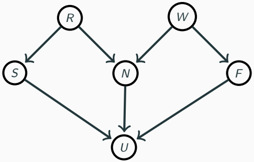
SAMPLING



simple forward sampling

- start sampling parents

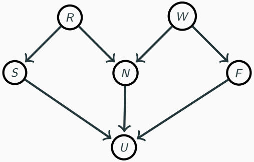
SAMPLING

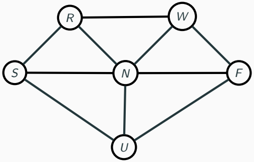


simple forward sampling

- start sampling parents
- sample through step by step

SAMPLING



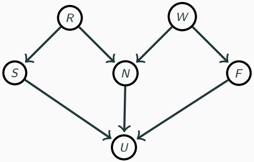


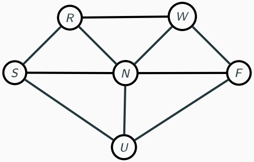
simple forward sampling

- start sampling parents
- sample through step by step

Gibbs sampling

SAMPLING





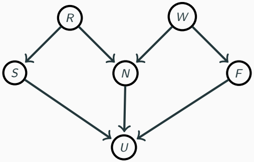
simple forward sampling

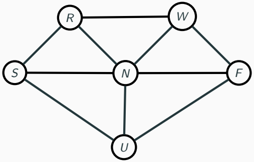
- start sampling parents
- sample through step by step

Gibbs sampling

- initialize values for all variables

SAMPLING





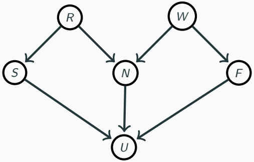
simple forward sampling

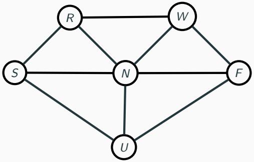
- start sampling parents
- sample through step by step

Gibbs sampling

- initialize values for all variables
- sample new values from full conditionals $P(X_i|\mathbf{X}_{-i})$

SAMPLING





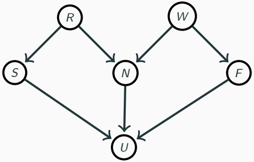
simple forward sampling

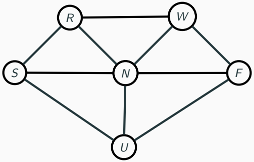
- start sampling parents
- sample through step by step

Gibbs sampling

- initialize values for all variables
- sample new values from full conditionals $P(X_i|\mathbf{X}_{-i})$
- use those values again and form a loop

SAMPLING





simple forward sampling

- start sampling parents
- sample through step by step

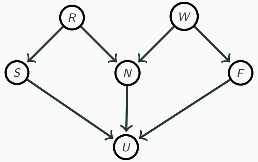
Gibbs sampling

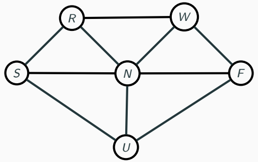
- initialize values for all variables
- sample new values from full conditionals $P(X_i|\mathbf{X}_{-i})$
- use those values again and form a loop

problems

- low p needs many samples (chernoff bound)

SAMPLING





simple forward sampling

- start sampling parents
- sample through step by step

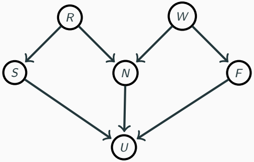
Gibbs sampling

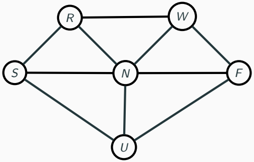
- initialize values for all variables
- sample new values from full conditionals $P(X_i|\mathbf{X}_{-i})$
- use those values again and form a loop

problems

- low p needs many samples (chernoff bound)
- evidence reduces acceptable samples

SAMPLING





simple forward sampling

- start sampling parents
- sample through step by step

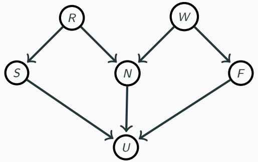
Gibbs sampling

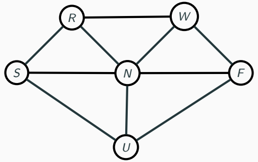
- initialize values for all variables
- sample new values from full conditionals $P(X_i|\mathbf{X}_{-i})$
- use those values again and form a loop

problems

- low p needs many samples (chernoff bound)
- evidence reduces acceptable samples
- when has it mixed?

SAMPLING





simple forward sampling

- start sampling parents
- sample through step by step

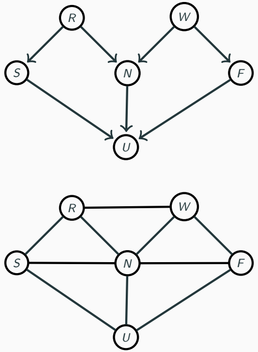
Gibbs sampling

- initialize values for all variables
- sample new values from full conditionals $P(X_i|\mathbf{X}_{-i})$
- use those values again and form a loop

problems

- low p needs many samples (chernoff bound)
- evidence reduces acceptable samples
- when has it mixed?
- are adjacent samples i.i.d.?

SAMPLING



simple forward sampling

- start sampling parents
- sample through step by step

Gibbs sampling

- initialize values for all variables
- sample new values from full conditionals $P(X_i|\mathbf{X}_{-i})$
- use those values again and form a loop

problems

- low p needs many samples (chernoff bound)
- evidence reduces acceptable samples
- when has it mixed?
- are adjacent samples i.i.d.?

REFERENCES

1. D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, MIT Press 2009.
2. D. Koller, *Probabilistic Graphical Models*. `https://class.coursera.org/pgm/lecture`

THANK YOU FOR YOUR ATTENTION

Questions?

VARIABLE ELIMINATION - COMPLEXITY

total operations linear in N, m and n

$\sum_{k=1}^{K} o_k^{(prod)} + o_k^{(sum)}$

VARIABLE ELIMINATION - COMPLEXITY

total operations linear in N, m and n

$\sum_{k=1}^{K} o_k^{(prod)} + o_k^{(sum)} =$

$\sum_{k=1}^{K}[(m_k - 1)N_k + \frac{d_k - 1}{d_k} N_k]$

VARIABLE ELIMINATION - COMPLEXITY

total operations linear in N, m and n

$\sum_{k=1}^{K} o_k^{(prod)} + o_k^{(sum)} =$

$\sum_{k=1}^{K}[(m_k - 1)N_k + \frac{d_k - 1}{d_k} N_k] < \sum_{k=1}^{K}[(m_k - 1)N + N]$

VARIABLE ELIMINATION - COMPLEXITY

total operations linear in N, m and n

$\sum_{k=1}^{K} o_k^{(prod)} + o_k^{(sum)} =$

$\sum_{k=1}^{K}[(m_k - 1)N_k + \frac{d_k - 1}{d_k} N_k] < \sum_{k=1}^{K}[(m_k - 1)N + N] < (m + n)N$

VARIABLE ELIMINATION - COMPLEXITY

total operations linear in N, m and n

$\sum_{k=1}^{K} o_k^{(prod)} + o_k^{(sum)} =$

$\sum_{k=1}^{K}[(m_k - 1)N_k + \frac{d_k - 1}{d_k} N_k] < \sum_{k=1}^{K}[(m_k - 1)N + N] < (m + n)N$

- r_k # variables in Ψ_k

VARIABLE ELIMINATION - COMPLEXITY

total operations linear in N, m and n

$\sum_{k=1}^{K} o_k^{(prod)} + o_k^{(sum)} =$

$\sum_{k=1}^{K}[(m_k - 1)N_k + \frac{d_k - 1}{d_k} N_k] < \sum_{k=1}^{K}[(m_k - 1)N + N] < (m + n)N$

- r_k # variables in Ψ_k
- $r := \max_k r_k \rightarrow$

VARIABLE ELIMINATION - COMPLEXITY

total operations linear in N, m and n

$\sum_{k=1}^{K} o_k^{(prod)} + o_k^{(sum)} =$

$\sum_{k=1}^{K}[(m_k - 1)N_k + \frac{d_k - 1}{d_k}N_k] < \sum_{k=1}^{K}[(m_k - 1)N + N] < (m + n)N$

- r_k # variables in Ψ_k
- $r := \max_k r_k \rightarrow$ **width** $+ 1$

VARIABLE ELIMINATION - COMPLEXITY

total operations linear in N, m and n

$\sum_{k=1}^{K} o_k^{(prod)} + o_k^{(sum)} =$

$\sum_{k=1}^{K}[(m_k - 1)N_k + \frac{d_k - 1}{d_k} N_k] < \sum_{k=1}^{K}[(m_k - 1)N + N] < (m + n)N$

- r_k # variables in Ψ_k
- $r :$ $= \max_k r_k \rightarrow$ **width** $+ 1$
- $d :$ $= \max_k d_k \rightarrow N \leq d^k$

VARIABLE ELIMINATION - COMPLEXITY

total operations linear in N, m and n

$\sum_{k=1}^{K} o_k^{(prod)} + o_k^{(sum)} =$

$\sum_{k=1}^{K}[(m_k - 1)N_k + \frac{d_k - 1}{d_k} N_k] < \sum_{k=1}^{K}[(m_k - 1)N + N] < (m + n)N$

- r_k # variables in Ψ_k
- $r := \max_k r_k \rightarrow$ **width** $+ 1$
- $d := \max_k d_k \rightarrow N \leq d^k$

worst case operations $(m + n)d^r$ exponential in r

THEME

For this presentation the 'Metropolis' theme by Matthias Vogelgesang (based on the 'hsrm' theme by Benjamin Weiss) was used.

Get the source of this theme and the demo presentation from:

`github.com/matze/mtheme`

The theme *itself* is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.