

概率图模型理论及应用

Theory and Applications of Probabilistic Graphical Models
(Lesson 3 – exact inference)

欧智坚

清华大学电子工程系

Addr: 罗姆楼 6-104

Tel: 62796193

Email: ozj@tsinghua.edu.cn

课程章节

❖ 第一章 图模型的表示理论 (2)

- Semantics (DGM, UGM)
- HMM, CRF

❖ 第二章 图模型的推理理论 (4)

- 精确推理: **variable-elimination, cluster-tree, triangulate**
- 连续变量: **Kalman**
- 采样近似: **sampling**
- 变分近似: **variational**

❖ 第三章 图模型的学习理论 (2)

- 参数学习: **maxlikelihoodEstimate, RFLearning, BayesEstimate**
- 结构学习: **StructureLearning**

	pgm-1 semantics ✓		pgm-2 hmm-crf ✓	pgm-4 kalman
			pgm-3 exact	pgm-5 sampling
pgm-6 variational	pgm-8 Bayesian			
pgm-7 ML				

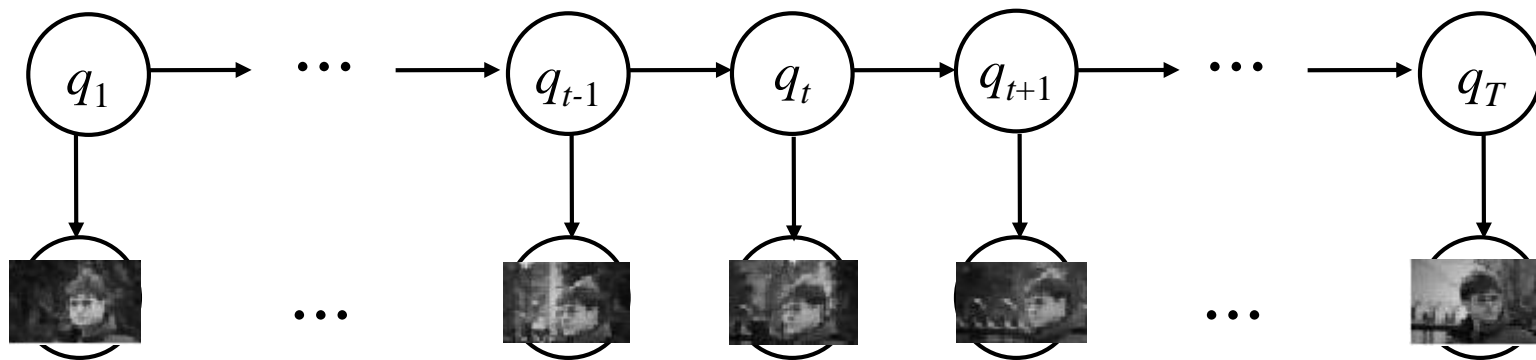
第三章 图模型的推理理论

推理计算

❖ **定义** 假设联合分布已知

给定观测量取值 $X_E=x_E$ ，计算其它一些未知量 X_Q 的条件分布

■ 即计算 $p(x_Q | X_E=x_E)$ e.g. $p(q_t | y_{1:T}) = ?$



$$p(x_Q | x_E) \propto p(x_Q, x_E) = \sum_{x_{V \setminus (Q,E)}} p(x_Q, x_E, x_{V \setminus (Q,E)})$$

$$p(q_t | y_{1:T}) \propto p(q_t, y_{1:T}) = \sum_{q_{1:T} \setminus \{t\}} p(q_{1:T}, y_{1:T}) = \sum_{q_{1:T} \setminus \{t\}} p(q_1) \cdot \prod_{t=1}^{T-1} p(q_{t+1} | q_t) \cdot \prod_{t=1}^T p(y_t | q_t)$$

Sum-product: 对连乘积求和

$q_{1:T} \setminus \{t\}$

$q_{1:T} \setminus \{t\}$

(π, A, B)

推理计算的三种具体形式

- ❖ 信度更新 (Belief updating)

$$p(x_Q | x_E) \propto \sum_{x_{V \setminus (Q,E)}} p(x_V) \quad \text{e.g. } p(q_t | y_{1:T}), p(q_t, q_{t+1} | y_{1:T}) \quad \text{Sum-product}$$

- ❖ 计算最有可能取值 (most probable explanation, MPE)

$$\begin{aligned} x_Q^* &= \arg \max_{x_Q} p(x_Q | x_E), \text{ where } Q \cup E = V \\ &= \arg \max_{x_Q} p(x_V) \quad \text{e.g. } \max_{q_{1:T}} p(q_{1:T} | y_{1:T}) \quad \text{Max-product} \end{aligned}$$

- ❖ 计算最大后验假设 (maximum a posteriori hypothesis, MAP)

$$\begin{aligned} x_Q^* &= \arg \max_{x_Q} p(x_Q | x_E), \text{ where } Q \cup E \subset V \\ &= \arg \max_{x_Q} \sum_{x_{V \setminus (Q,E)}} p(x_V) \quad \text{Max-sum-product} \end{aligned}$$

第三章 图模型的推理理论

推理计算的三大形式

Belief Updating

通过例子引出VE

For UGM

证据处理

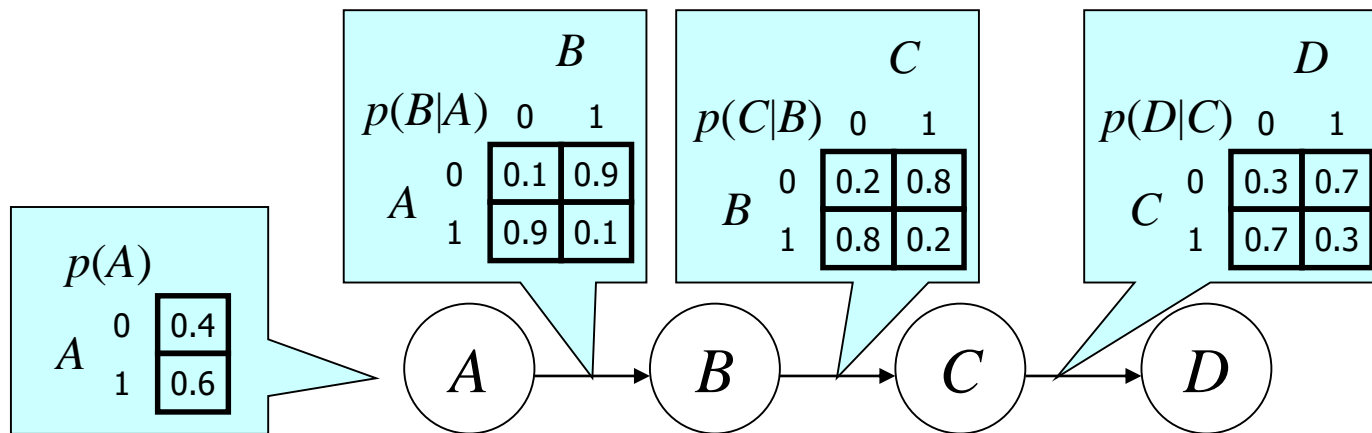
复杂度分析

MPE

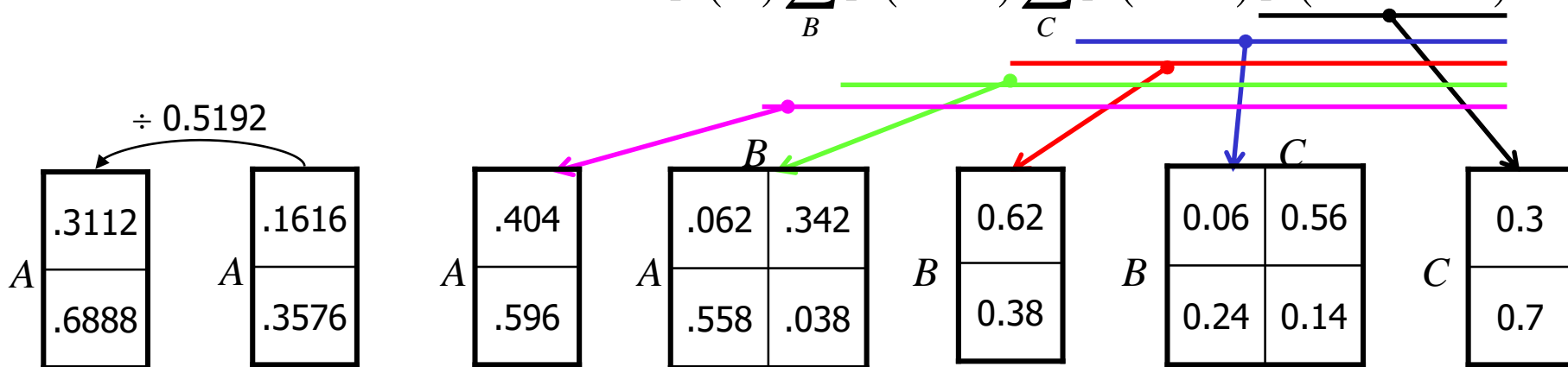
MAP

HMM-CRF中应用

Inference example



$$\begin{aligned}
 p(A | D=0) &\propto p(A, D=0) = \sum_B \sum_C p(A) p(B|A) p(C|B) p(D=0|C) \\
 &= p(A) \sum_B p(B|A) \sum_C p(C|B) p(D=0|C)
 \end{aligned}$$



Two basic operations: product, marginalization

Comment

- ❖ **Exponential reduction in computation !**
 - 对特定变量的求和尽可能往早挪，直到求和号里的乘积项均包含这个变量
 - 缓存中间结果（Cache intermediate results）

Variable elimination (bucket elimination):

对上面操作的一个系统的算法描述

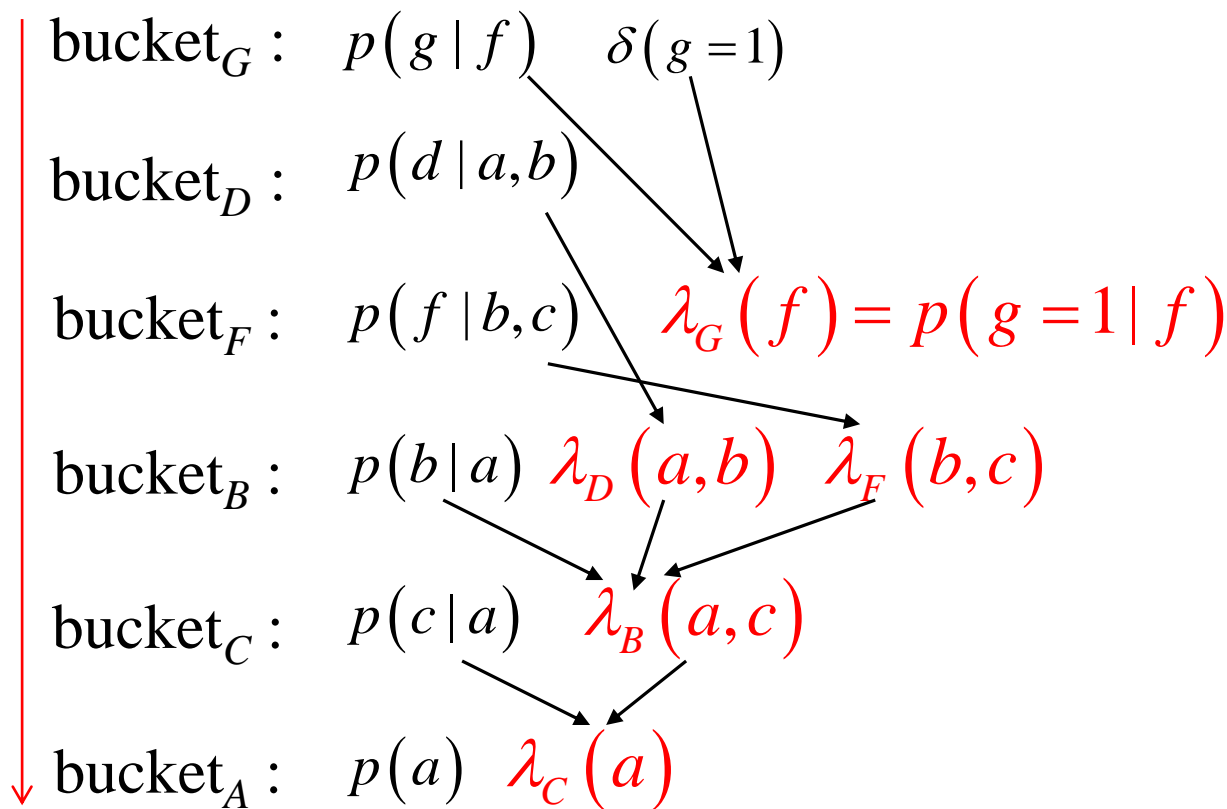
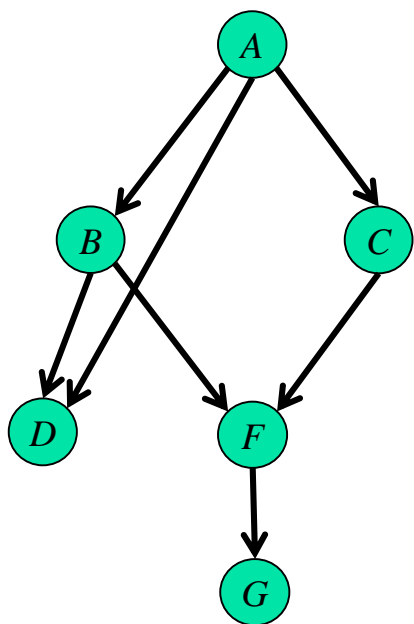
R. Dechter, "Bucket Elimination: A Unifying Framework for Probabilistic Inference", 1996

Bucket elimination *elim-bel* (Dechter 1996)

$$p(a | g = 1) \propto \sum_c \sum_b \sum_f \sum_d \sum_g p(a) p(c | a) p(b | a) p(f | b, c) p(d | a, b) p(g | f) \delta(g = 1)$$

给定变量消除排序 (例如 $\leftarrow a, c, b, f, d, g$)，询问结点居首

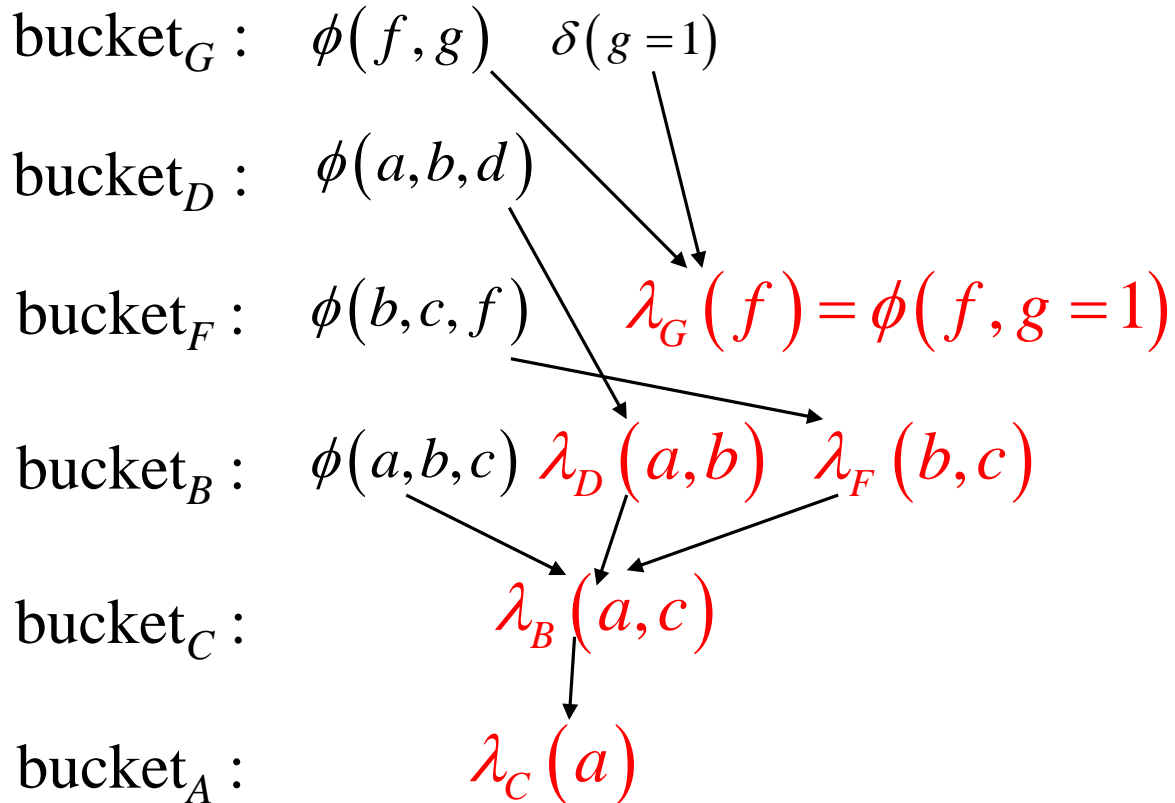
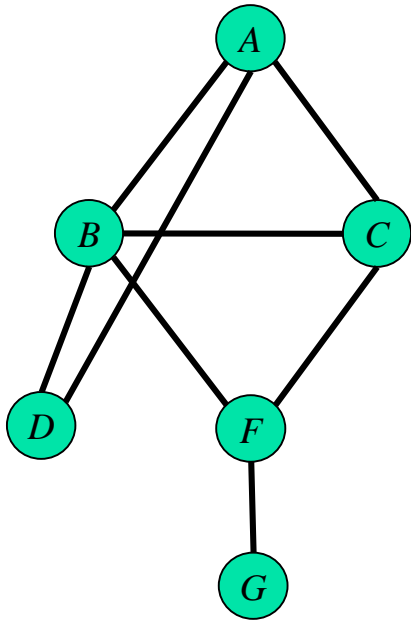
- ① **初始化**: 将局部函数放入桶中。规则: 最先被消除的自变量所标识的桶
- ② **依次消除变量**: 桶内各函数相乘, 消除掉本变量, 消息函数放入新桶
- ③ **返回**: 询问结点标识的桶中所有函数相乘



Bucket elimination for UGM

$$p(a,b,c,d,f,g) = \frac{1}{Z} \phi(a,b,c) \phi(b,c,f) \phi(a,b,d) \phi(f,g)$$

$$p(a | g=1) \propto \sum_c \sum_b \sum_f \sum_d \sum_g \phi(a,b,c) \phi(b,c,f) \phi(a,b,d) \phi(f,g) \delta(g=1)$$

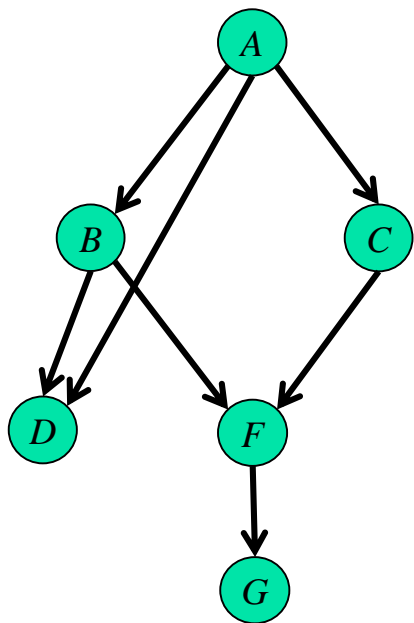


尽早代入证据

$$p(a|b=1) \propto \sum_c \sum_b \sum_f \sum_d \sum_g p(a) p(c|a) p(b|a) p(f|b,c) p(d|a,b) p(g|f) \delta(b=1)$$

最先消去证据变量，或者说，直接代入

$$p(a|b=1) \propto \sum_c \sum_f \sum_d \sum_g p(a) p(c|a) p(b=1|a) p(f|b=1,c) p(d|a,b=1) p(g|f)$$



$$\text{bucket}_G : p(g|f)$$

$$\text{bucket}_D : p(d|a,b)$$

$$\text{bucket}_F : p(f|b,c) \quad \lambda_G(f)$$

$$\text{bucket}_B : p(b|a) \quad \lambda_D(a,b) \quad \lambda_F(b,c) \quad \delta(b=1)$$

$$\text{bucket}_C : p(c|a) \quad \lambda_B(a,c)$$

$$\text{bucket}_A : p(a) \quad \lambda_C(a)$$

复杂度分析

- ❖ Complexities (time and space) vary greatly for different elimination ordering.
- ❖ Complexity of processing a bucket

$$\lambda_i(\cdot) = \sum_{x_i} \prod_{h \in \text{bucket}_i} h \quad \text{scope}(\lambda_i) = \left[\bigcup_{h \in \text{bucket}_i} \text{scope}(h) \right] - \{X_i\}$$

$$\lambda_B(a, c) = \sum_b p(b|a) \lambda_D(a, b) \lambda_F(b, c)$$

消息函数的
自变量集的大小

- Time complexity for computing message functions

$$r^{|\text{scope}(\lambda_i)|+1}$$

- Space complexity for saving message functions

$$r^{|\text{scope}(\lambda_i)|}$$

To be determined by graph-theoretic analysis ?

连乘积的素图表示(primal graph)

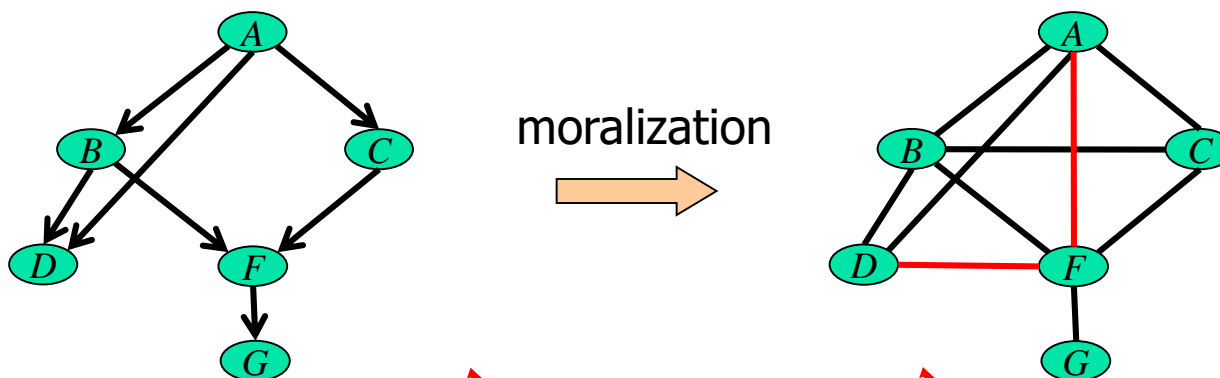
❖ 一个连乘积的素图表示如下定义：

- 以变量为结点；
- 属于同一个局部函数的变量，两两之间添加（无向）边。

❖ 举例

- 一个无向图的分布的素图表示是：无向图自身
- 一个有向图的分布的素图表示是：有向图的moral graph
- 代入证据后的连乘积（e.g. $c=1$ ）
- 连乘积求和

$$p(a,b,c,d,f,g) \propto \phi(a,b,c)\phi(b,c,f)\phi(a,b,d)\phi(f,g)$$



$$p(a,b,c,d,f,g) = p(a) p(c|a) p(b|a) p(f|b,c) p(d|a,b) p(g|f)$$

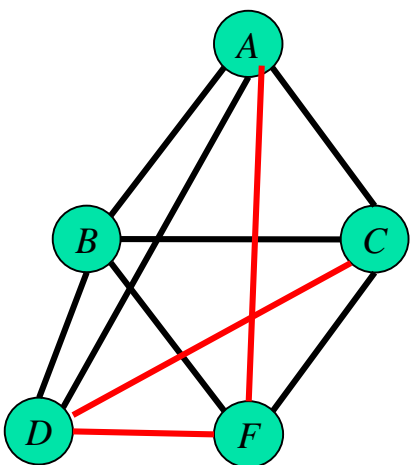
Graphic effect of variable elimination

连乘积消除一个变量，仍是一个（新的）连乘积

消除变量 x_i ：图上消除结点 x_i 且将 x_i 的邻居结点两两间加上边

消除 X_i 时, $|\text{scope}(\lambda_i)| = X_i$ 的邻居数目

$$p(a | g = 1) \propto \sum_f \sum_d \sum_c \sum_b p(a) p(c | a) p(b | a) p(f | b, c) p(d | a, b) p(g = 1 | f)$$



$$\text{bucket}_B : p(d | a, b) \quad p(f | b, c) \quad p(b | a)$$

$$\text{bucket}_C : p(c | a) \quad \lambda_B(a, c, d, f)$$

$$\text{bucket}_D : \lambda_C(a, d, f)$$

$$\text{bucket}_E : p(g = 1 | f) \quad \lambda_F(a, f)$$

$$\text{bucket}_A : p(a) \quad \lambda_F(a)$$

4
3
2
1

$$|\text{scope}(\lambda_i)|_4$$

Induced-graph, induced-width

- ❖ 一个素图 G 在结点排序 $d=(x_1, \dots, x_N)$ 下的诱导图 G_d 如下定义：
 - 从后往前依次消除各结点；
 - 消除结点 x_i 时，将此时图中 x_i 的邻居两两间添加边。
 - 称此时图中 x_i 的邻居数目为：结点 x_i 在排序 d 下的**诱导宽度**
 - $G_d = G$ 加上上述过程多添加的边。 = 消除 x_i 得到的消息函数的自变量集大小
- ❖ 图 G 在排序 d 下的**诱导宽度** $w(G_d)$
= 图 G 的各结点在排序 d 下的诱导宽度的**最大者**
- ❖ 图 G 的**诱导宽度** $w(G)$
= 图 G 在所有可能排序下的诱导宽度的**最小者**

Complexity theorem

- ❖ Given an ordering $d=(X_1, \dots, X_N)$ of a primal graph G , the time complexity of *elim-bel* is $O(N \cdot r^{w(G_d)+1})$ and space complexity is $O(N \cdot r^{w(G_d)})$.
where r is the maximum domain size of any variable.
- ❖ Finding an ordering with the smallest induced-width is NP-hard.
 - Useful greedy heuristics are available.
 - 人为在图上找出好的变量排序。

Comment

- ❖ **Exponential reduction in computation !**
 - 对特定变量的求和尽可能往早挪，直到求和号里的乘积项均包含这个变量 ?
 - 缓存中间结果 (Cache intermediate results)

Variable elimination (bucket elimination):

对上面操作的一个系统的算法描述

R. Dechter, "Bucket Elimination: A Unifying Framework for Probabilistic Inference", 1996

S. M. Aji and R. J. McEliece, "The generalized distributive law,"
IEEE Trans. Information Theory, vol.46, pp.325-343, Mar. 2000.

第三章 图模型的推理理论

推理计算的三大形式

Belief Updating

通过例子引出VE

For UGM

证据处理

复杂度分析

MPE

MAP

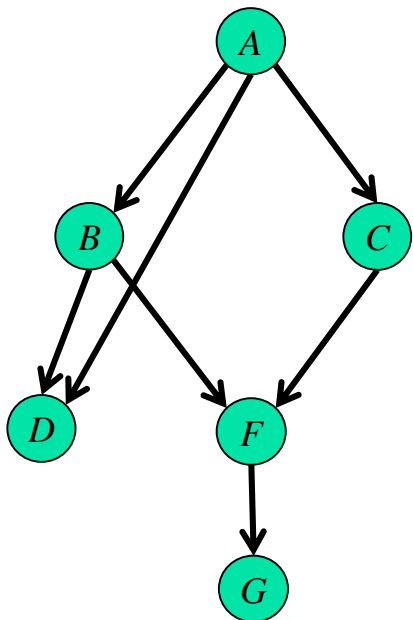
HMM-CRF中应用

Finding MPE *elim-mpe* (Dechter 1996)

$$MPE = \arg \max_{a,c,b,f,d} p(a,c,b,f,d \mid g=1)$$

$$\max_{a,c,b,f,d} p(a) p(c|a) p(b|a) p(f|b,c) p(d|a,b) p(g=1|f)$$

Σ is replaced by max



$$D: \max_d p(d|a,b)$$

$$F: p(f|b,c) p(g=1|f)$$

$$B: p(b|a) \lambda_D(a,b) \lambda_F(b,c) \quad D^*(a,b) = \arg \max_d p(d|a,b)$$

$$C: p(c|a) \lambda_B(a,c) \quad F^*(b,c) = \arg \max_f p(f|b,c) p(g=1|f)$$

$$A: p(a) \lambda_C(a) \quad B^*(a,c) = \arg \max_b p(b|a) \lambda_D(a,b) \lambda_F(b,c)$$

$$C^*(a) = \arg \max_c p(c|a) \lambda_B(a,c)$$

$$A^* = \arg \max_a p(a) \lambda_C(a)$$

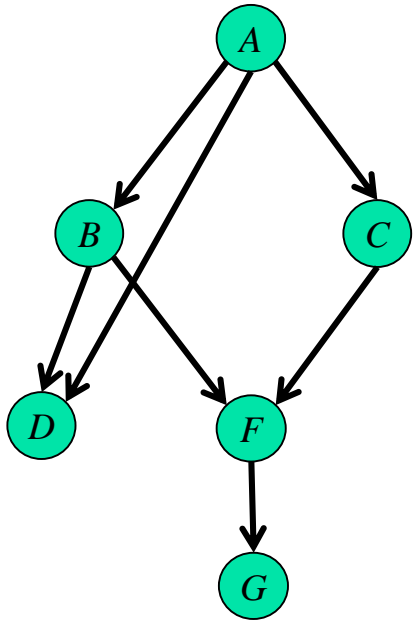
MPE

Finding MAP *elim-map* (Dechter 1996)

Elimination operation: either summation or maximization

Restriction on variable ordering: summation before maximization

$$\max_{a,c} p(a,c | g = 1) = \max_{a,c} \sum_{b,f,d} p(a) p(c|a) p(b|a) p(f|b,c) p(d|a,b) p(g=1|f)$$



$$D : p(d|a,b)$$

$$F : p(f|b,c) \quad p(g=1|f)$$

$$B : p(b|a) \quad \lambda_D(a,b) \quad \lambda_F(b,c)$$

$$C : p(c|a) \quad \lambda_B(a,c)$$

$$A : p(a) \quad \lambda_C(a) \quad C^*(a) = \arg \max_c p(c|a) \lambda_B(a,c)$$

$$A^* = \arg \max_a p(a) \lambda_C(a)$$

MAP

第三章 图模型的推理理论

推理计算的三大形式

Belief Updating

通过例子引出VE

For UGM

证据处理

复杂度分析

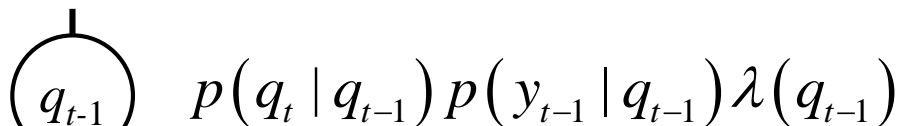
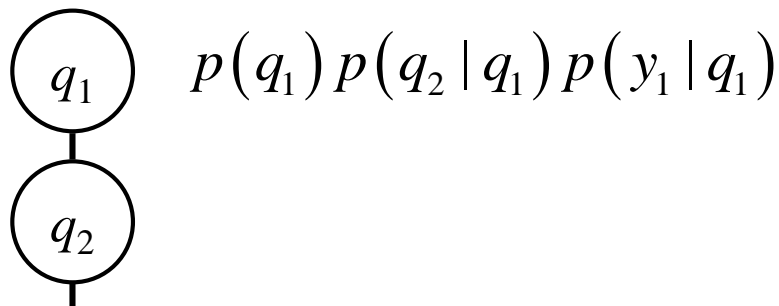
MPE

MAP

HMM-CRF中应用

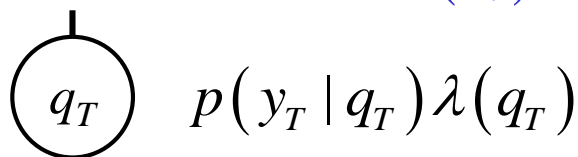
Viterbi algorithm: MPE

$$\max_{q_{1:T}} p(q_{1:T} | y_{1:T}) \propto \max_{q_T \cdots q_1} p(q_1) \cdot \prod_{t=1}^{T-1} p(q_{t+1} | q_t) \cdot \prod_{t=1}^T p(y_t | q_t)$$



$$\lambda(q_t) = \max_{q_{t-1}} p(q_t | q_{t-1}) \underbrace{p(y_{t-1} | q_{t-1}) \lambda(q_{t-1})}$$

$$\delta(q_t) = p(y_t | q_t) \max_{q_{t-1}} p(q_t | q_{t-1}) \delta(q_{t-1})$$



Forward-backward algorithm

$$q_1 \quad p(q_1) p(q_2 | q_1) p(y_1 | q_1)$$

$$q_2$$

$$p(q_t | y_{1:T}) \propto \sum_{q_{1:T} \setminus q_t} p(q_1) \cdot \prod_{t=1}^{T-1} p(q_{t+1} | q_t) \cdot \prod_{t=1}^T p(y_t | q_t)$$

$$p(q_t | y_{1:T}) \propto \underbrace{p(y_t | q_t) \lambda(q_t)}_{\alpha(q_t)} \cdot \beta(q_t)$$

$$q_{t-1}$$

$$p(q_t | q_{t-1}) p(y_{t-1} | q_{t-1}) \lambda(q_{t-1})$$

$$q_t$$

$$p(y_t | q_t) \lambda(q_t) \quad \beta(q_t) = \sum_{q_{t+1}} p(q_{t+1} | q_t) p(y_{t+1} | q_{t+1}) \beta(q_{t+1})$$

$$q_{t+1}$$

$$p(q_{t+1} | q_t) p(y_{t+1} | q_{t+1}) \beta(q_{t+1})$$

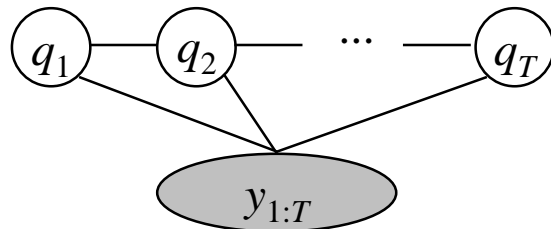
$$q_{T-1}$$

$$p(q_{T-1} | q_{T-2}) p(y_{T-1} | q_{T-1}) \beta(q_{T-1})$$

$$q_T$$

$$p(q_T | q_{T-1}) p(y_T | q_T)$$

Linear-chain CRF



q_t, q_{t+1} 组成的簇上的

特征函数 $\psi_t(q_t, q_{t+1}, y)$

q_t 组成的簇上的

特征函数 $\psi_t(q_t, y)$

$$p(q_{1:T} | y) \propto \exp \left\{ \sum_{t=1}^{T-1} \sum_i \lambda_i f_i(q_t, q_{t+1}, y, t) + \sum_{t=1}^T \sum_j \mu_j f_j(q_t, y, t) \right\}$$

Likelihood calculation——Forward-backward算法

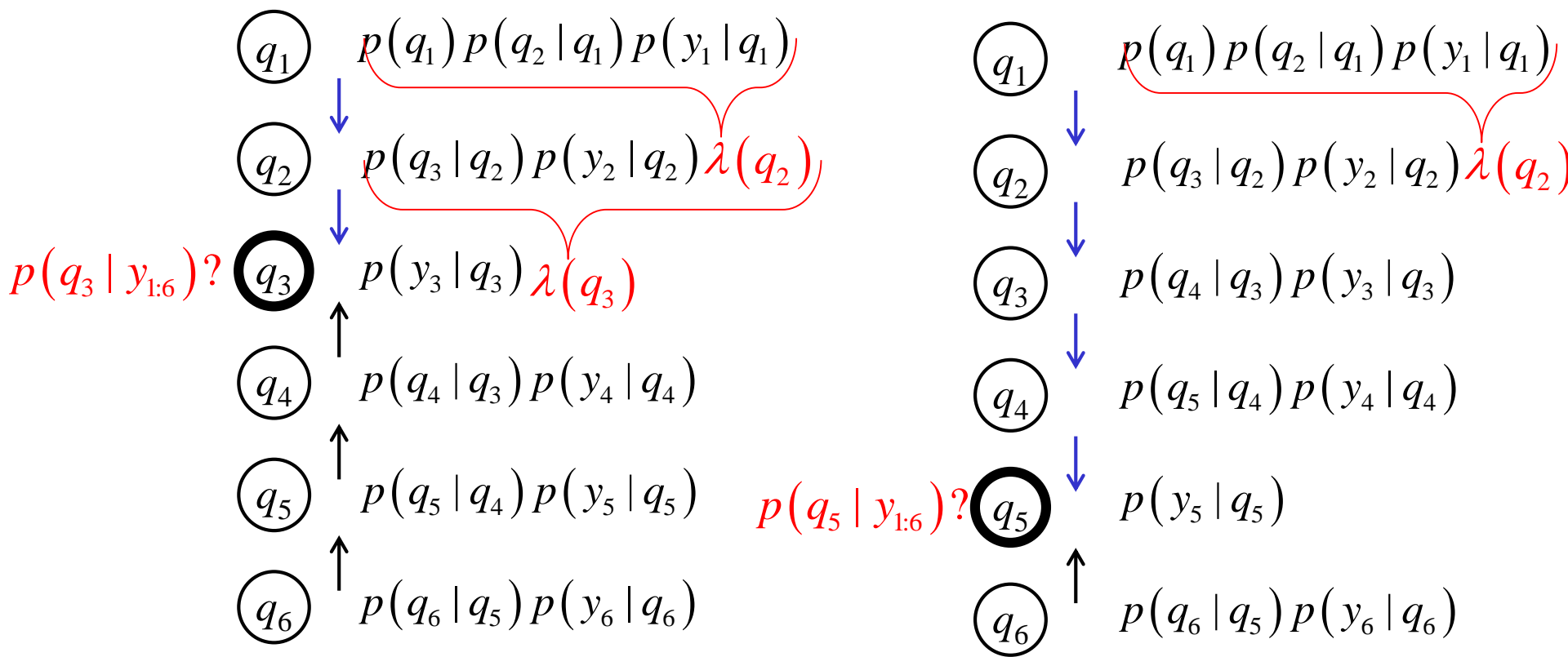
$$p(q_t | y) \propto \sum_{q_{1:T} \setminus q_t} \exp \left\{ \sum_{t=1}^{T-1} \psi_t(q_t, q_{t+1}, y) + \sum_{t=1}^T \psi_t(q_t, y) \right\}$$

Decoding (recognition)——Viterbi算法

$$\max_{q_{1:T}} p(q_{1:T} | y) \propto \max_{q_T \cdots q_1} \exp \left\{ \sum_{t=1}^{T-1} \psi_t(q_t, q_{t+1}, y) + \sum_{t=1}^T \psi_t(q_t, y) \right\}$$

多遍运行 Variable elimination

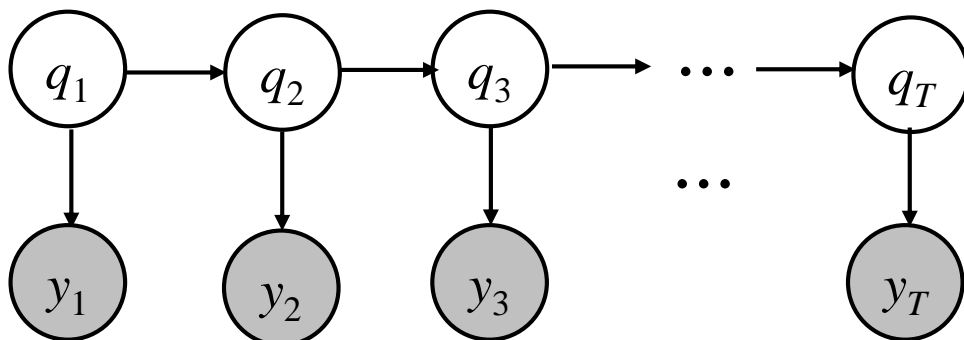
- 求多个隐变量的后验分布？ $p(q_t | y_{1:T}), \forall t$



5次 α 计算、5次 β 计算 VS 5次消息计算/遍 * 6遍ve

Limitations of VE

- ❖ How to avoid **redundant computation** for multiple runs of elimination?



Consider for the above graph, we need to compute $p(q_1 | y_1, y_2, \dots, y_T)$, $p(q_2 | y_1, y_2, \dots, y_T)$, $p(q_3 | y_1, y_2, \dots, y_T)$, ...

Go to **cluster-tree algorithm**

- ❖ If $w(G)$ is too large ?

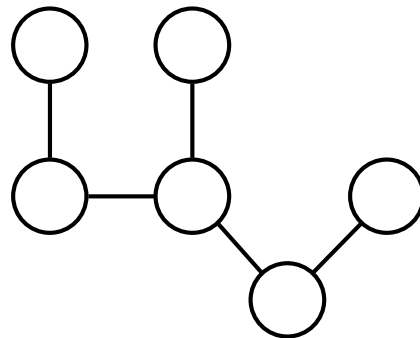
The variable-elimination algorithm is of no help here; the complexity is **intrinsic to the graph**.

Go to **approximate algorithm**

Cluster-tree Elimination (树消除算法)

Basic concepts – tree, leaf

A graph G is connected		图上任两个结点间都有一条迹
tree	T	如果一个图是连通的，并且没有环
Leaf of a tree		树上只有一个邻居的结点

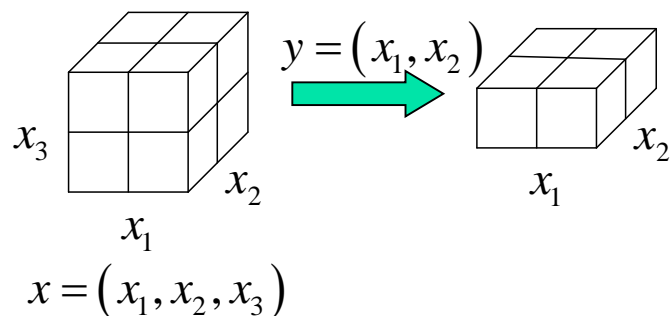


Basic concepts – Marginal

- ❖ 假设 x 和 y 为变量集，且 $y \subseteq x$ 。 $\phi_X(x)$ 为定义在 x 上的函数，则 $\phi_X(x)$ 在 y 上的边缘化为

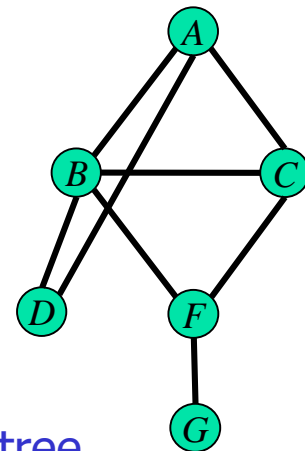
$$\Downarrow_y \phi_X(x) = \sum_{x \setminus y} \phi_X(x)$$

$$\Downarrow_y \phi_X(x) = \max_{x \setminus y} \phi_X(x)$$



引入 tree-decomposition(树分解)/cluster-tree

$$p(a | g = 1) \propto \sum_c \sum_b \sum_f \sum_d \sum_g \phi(a, b, c) \phi(b, c, f) \phi(a, b, d) \phi(f, g) \delta(g = 1)$$



用一个树型结构来组织 **Variable Elimination** 的计算。

给定变量消除排序，询问结点 x_Q （这里即 A 结点）居首。

每个桶视为一个 **cluster**，cluster 上贴变量集标签

① 初始化：

将连乘积中的局部函数放入 cluster 中

② 依次消息计算（从叶子到根）：

以 x_Q 对应 cluster 为根；

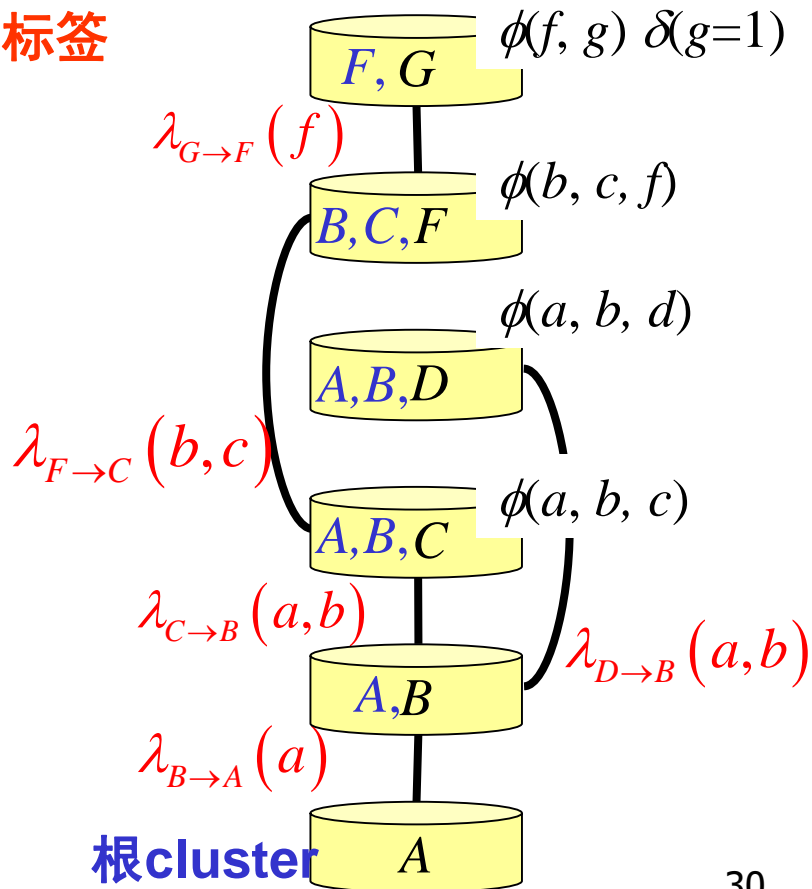
产生的消息函数与边关联。

处理一个桶/一次消息计算：

初始放置在桶中的局部函数 与 先后放到桶中的消息函数相乘，再消除变量，得到一个新的消息函数

③ 返回 $p(x_Q | e)$ ：

bucket-tree



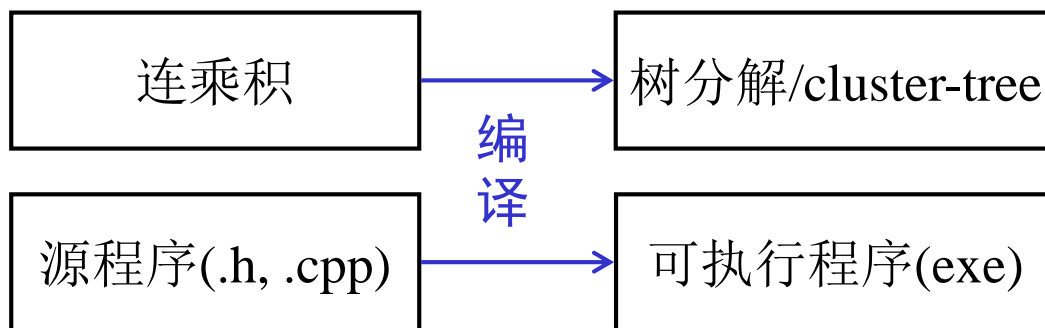
观念的转变

❖ Variable elimination（变量消除算法）

- 一边消息计算，一边生成一棵树

❖ Cluster-tree elimination（树消除算法）

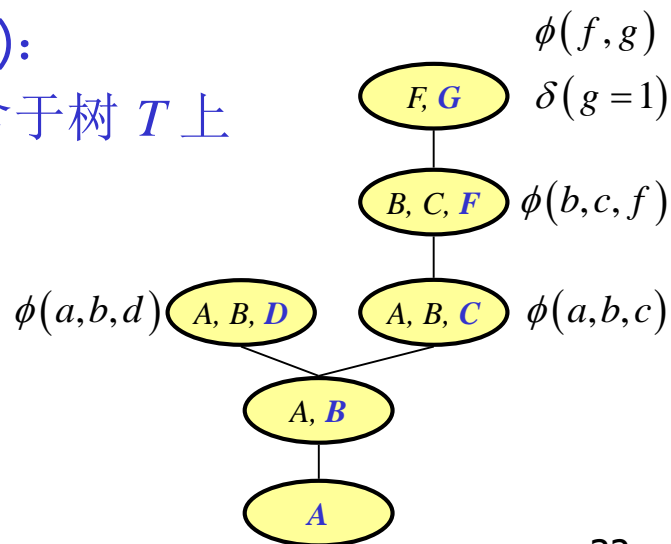
- 事先编译生成cluster-tree，然后执行计算
- 把 编译（寻找最优的cluster-tree）与 计算 分离



tree-decomposition/cluster-tree 定义

- ❖ 一个连乘积 $G: \prod_i f_i$ 的树分解: 一个三元组 $\langle T, \chi, \psi \rangle$, 其中
 - G 中每个乘积项 (局部函数) f_i 是一个多变量实值非负函数
 - T 是一棵树, 称 树上每个结点 C 为 cluster
 - 每个 cluster 上贴有一个变量集 **标签** $\chi(C)$ 。
 - 每个 cluster 内装有一组函数 $\psi(C)$ 。对每个局部函数 f_i , 有且仅有一个 cluster C 使得 $f_i \in \psi(C)$ 。所有放置在一个 cluster C 的局部函数的变量域的并 $\subseteq \chi(C)$
 - 树 T 满足 **running intersection property (RIP)**:
树 T 上任两个 cluster 的交 $\chi(C_1) \cap \chi(C_2)$ 均包含于树 T 上 C_1 与 C_2 之间路径上的每一个 cluster 的变量集

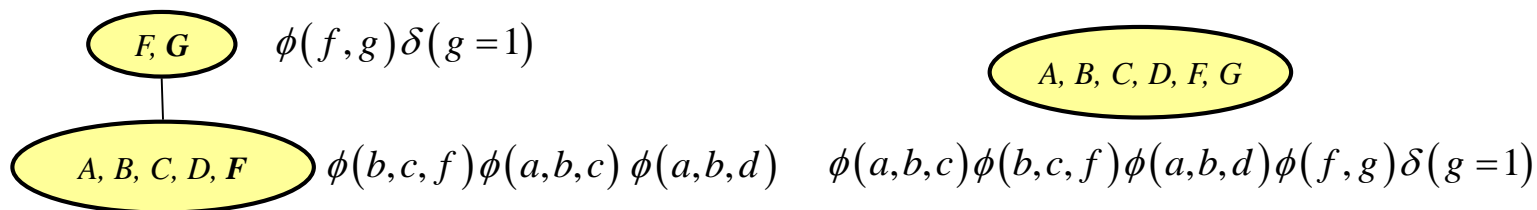
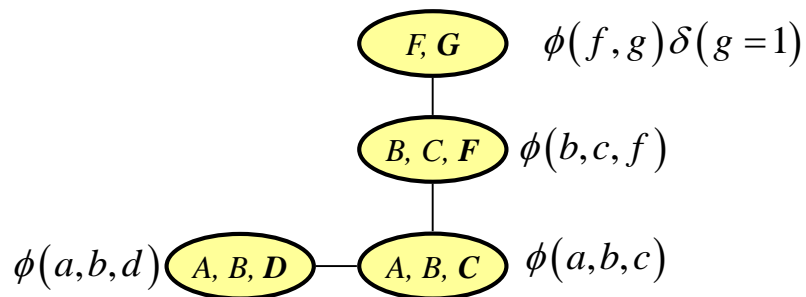
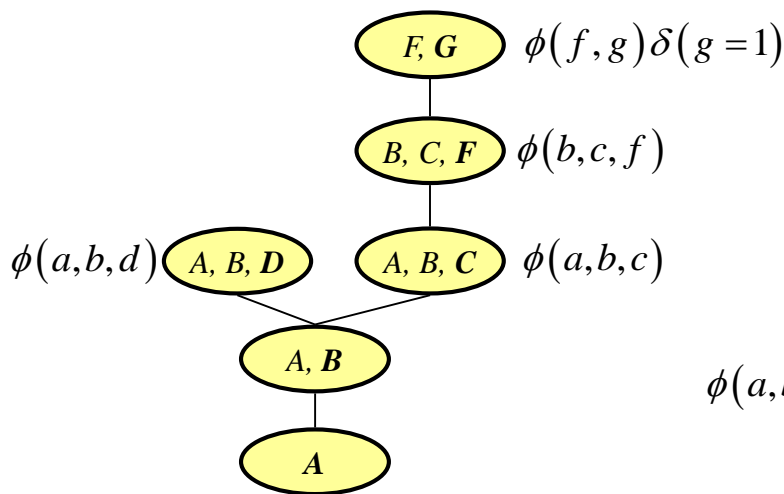
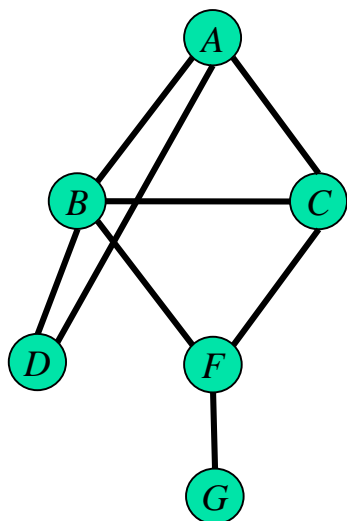
连乘积 G : $\phi(a,b,c)\phi(b,c,f)\phi(a,b,d)\phi(f,g)\delta(g=1)$



tree-decomposition — example

一个连乘积的不同树分解

连乘积 G : $\phi(a,b,c)\phi(b,c,f)\phi(a,b,d)\phi(f,g)\delta(g=1)$



Cluster-tree elimination

❖ Consider cluster-tree as a computational data-structure

① 初始化:

给定连乘积 $\prod_i f_i$ 的一个树分解

② 依次消息计算（从叶子到根）:

任选一个cluster为根 C^{root} ，进行从叶子至根的消息计算

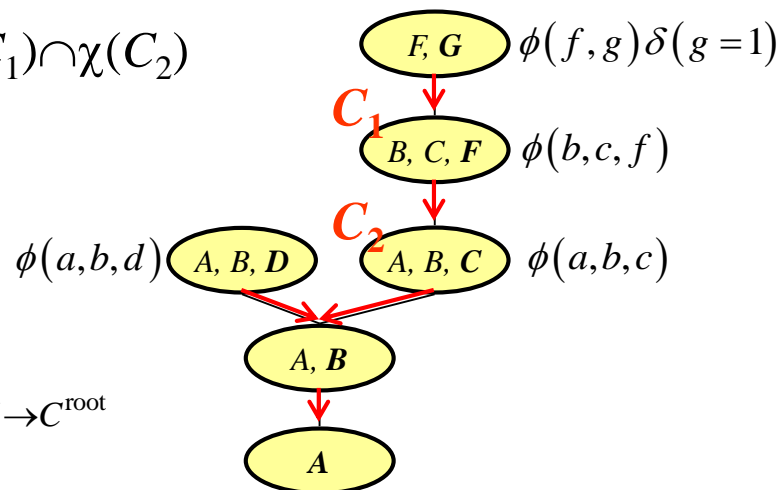
$$\lambda_{C_1 \rightarrow C_2} = \Downarrow_{\text{sep}(C_1, C_2)} \left\{ C_1 \text{ 里的函数} \cdot \prod_{Z \in C_1 \text{ 的邻居} \setminus C_2} \lambda_{Z \rightarrow C_1} \right\}$$

树 T 上两个相邻cluster的交记为 $\text{sep}(C_1, C_2) = \chi(C_1) \cap \chi(C_2)$

③ 返回:

连乘积在根cluster变量集 $\chi(C^{\text{root}})$ 上的边缘化

$$\left(\prod_i f_i \right) \Downarrow_{\chi(C^{\text{root}})} = C^{\text{root}} \text{ 里的函数} \cdot \prod_{Z \in C^{\text{root}} \text{ 的邻居}} \lambda_{Z \rightarrow C^{\text{root}}}$$



Cluster-tree elimination — schedule

❖ 求各个cluster的变量集的后验分布？

- Naïve: 以每个cluster为根，分别进行叶子至根的消息计算
- 高明: 只需计算出每条边 (C_1, C_2) 上的两个消息 $\lambda_{C_1 \rightarrow C_2}$ 和 $\lambda_{C_2 \rightarrow C_1}$
- 诸多消息计算的先后顺序？

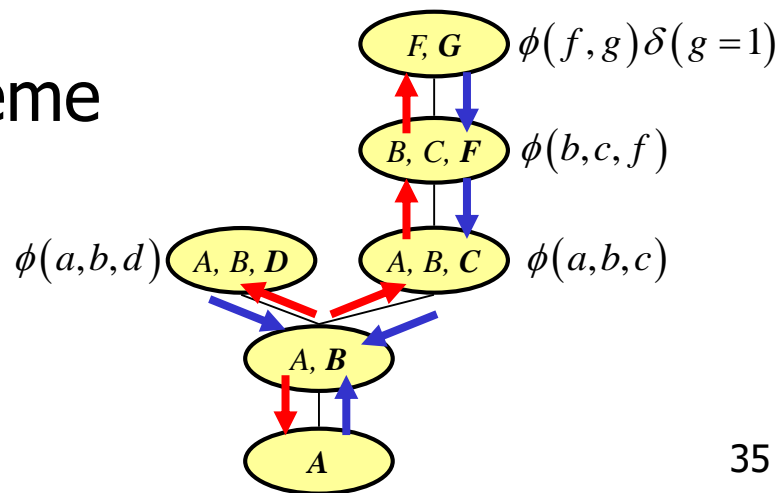
$$\lambda_{C_1 \rightarrow C_2} = \Downarrow_{sep(C_1, C_2)} \left\{ C_1 \text{ 里的函数} \cdot \prod_{Z \in C_1 \text{ 的邻居} \setminus C_2} \lambda_{Z \rightarrow C_1} \right\}$$

❖ Message protocol

- cluster C_1 可以发消息 $\lambda_{C_1 \rightarrow C_2}$ 给相邻cluster C_2 ，仅当“ C_1 的除 C_2 外的所有邻居”发送到 C_1 的消息都已计算好。

❖ A simple message passing scheme

- Leaf-to-root
- Root-to-leaf

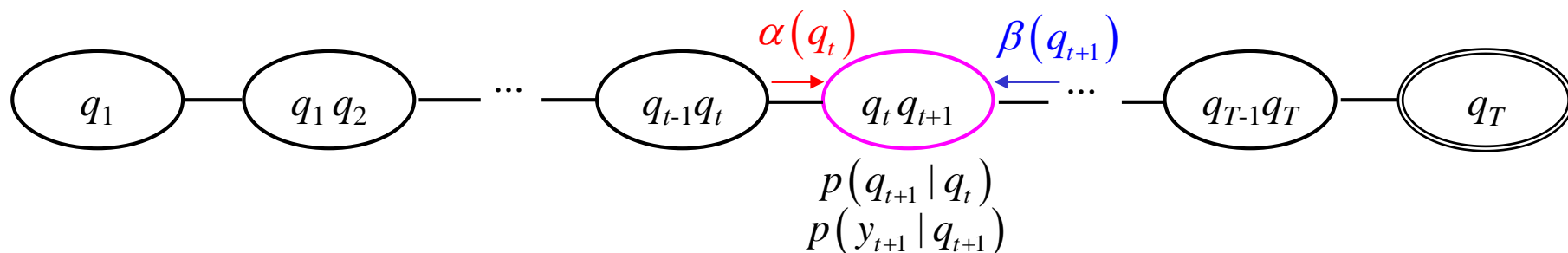


$$\left(\prod_i f_i \right) \Downarrow_{x(C)} = C \text{ 里的函数} \cdot \prod_{Z \in C \text{ 的邻居}} \lambda_{Z \rightarrow C}$$

举例：HMM's Forward-backward algorithm

连乘积： $p(q_1) \cdot \prod_{t=1}^{T-1} p(q_{t+1} | q_t) \cdot \prod_{t=1}^T p(y_t | q_t)$

求： $p(q_t, q_{t+1} | y_{1:T})$, $t = 1, \dots, T-1$ $\xi_t(i, j) = p(q_t = i, q_{t+1} = j | y_{1:T})$



$$\alpha(q_{t+1}) = p(y_{t+1} | q_{t+1}) \sum_{q_t=1}^N p(q_{t+1} | q_t) \alpha(q_t)$$

$$\beta(q_t) = \sum_{q_{t+1}=1}^N p(y_{t+1} | q_{t+1}) \beta(q_{t+1}) p(q_{t+1} | q_t)$$

$$p(q_t, q_{t+1} | y_{1:T}) \propto \alpha(q_t) \cdot \beta(q_{t+1}) \cdot p(q_{t+1} | q_t) p(y_{t+1} | q_{t+1})$$

Cluster-tree elimination — history

- ❖ Pearl's belief propagation (1988)
 - for single-root query in poly-tree (i.e. DAG forest, e.g. HMM)
- ❖ Shafer-Shenoy algorithm (1990)
 - Variable elimination
 - Probability propagation
- ❖ Hugin algorithm (1990)
 - F.V. Jensen, S. Lauritzen, and K. Olesen
 - Evolving from clique potentials to clique marginals
- ❖ Cluster-tree elimination (Dechter, 2005)
 - Unifying
- ❖ Sum-product algorithm (McEliece 1997, Frey 2001)
 - operates in factor graph
 - Loopy sum-product

课程章节

❖ 第一章 图模型的表示理论 (2)

- Semantics (DGM, UGM)
- HMM, CRF

❖ 第二章 图模型的推理理论 (4)

- 精确推理: **variable-elimination, cluster-tree, triangulate**
- 连续变量: **Kalman**
- 采样近似: **sampling**
- 变分近似: **variational**

❖ 第三章 图模型的学习理论 (2)

- 参数学习: **maxlikelihoodEstimate, RFLearning, BayesEstimate**
- 结构学习: **StructureLearning**

			pgm-2 hmm-crf ✓	pgm-4 kalman
	pgm-1 semantics ✓		pgm-3 exact ✓	pgm-5 sampling
pgm-6 variational	pgm-8 Bayesian			
pgm-7 ML				