Graphical models, message-passing algorithms, and variational methods: Part I

Martin Wainwright

Department of Statistics, and

Department of Electrical Engineering and Computer Science,

UC Berkeley, Berkeley, CA USA

Email: wainwrig@{stat,eecs}.berkeley.edu

For further information (tutorial slides, films of course lectures), see:

www.eecs.berkeley.edu/~wainwrig/

Introduction

- graphical models provide a rich framework for describing *large-scale multivariate statistical models*

- used and studied in many fields:
  - statistical physics
  - computer vision
  - machine learning
  - computational biology
  - communication and information theory
  - .....

- based on correspondences between graph theory and probability theory

Some broad questions

- **Representation:**
  - What phenomena can be captured by different classes of graphical models?
  - Link between graph structure and representational power?

- **Statistical issues:**
  - How to perform inference (data $\longrightarrow$ hidden phenomenon)?
  - How to fit parameters and choose between competing models?

- **Computation:**
  - How to structure computation so as to maximize efficiency?
  - Links between computational complexity and graph structure?

Outline

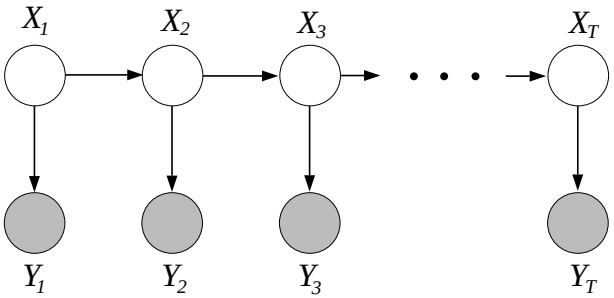
1. Background and set-up
   (a) Background on graphical models
   (b) Illustrative applications

2. Basics of graphical models
   (a) Classes of graphical models
   (b) Local factorization and Markov properties

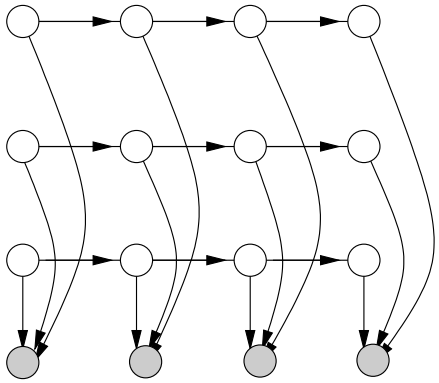
3. Exact message-passing on (junction) trees
   (a) Elimination algorithm
   (b) Sum-product and max-product on trees
   (c) Junction trees

4. Parameter estimation
   (a) Maximum likelihood
   (b) Proportional iterative fitting and related algorithsm
   (c) Expectation maximization

Example: Hidden Markov models



(a) Hidden Markov model



(b) Coupled HMM

- HMMs are widely used in various applications

  discrete X_t: computational biology, speech processing, etc.

  Gaussian X_t: control theory, signal processing, etc.

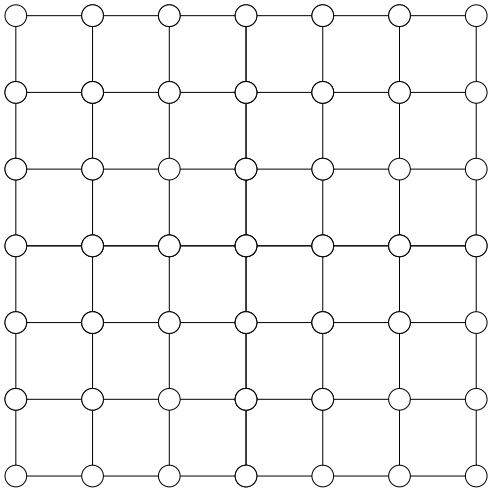
  coupled HMMs: fusion of video/audio streams

- frequently wish to solve *smoothing* problem of computing $p(x_t \mid y_1, \ldots, y_T)$

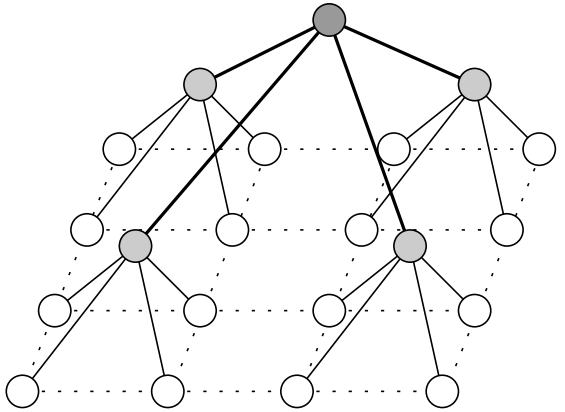
- exact computation in HMMs is tractable, but coupled HMMs require algorithms for approximate computation (e.g., structured mean field)

Example: Image processing and computer vision (I)

(a) Natural image



(b) Lattice



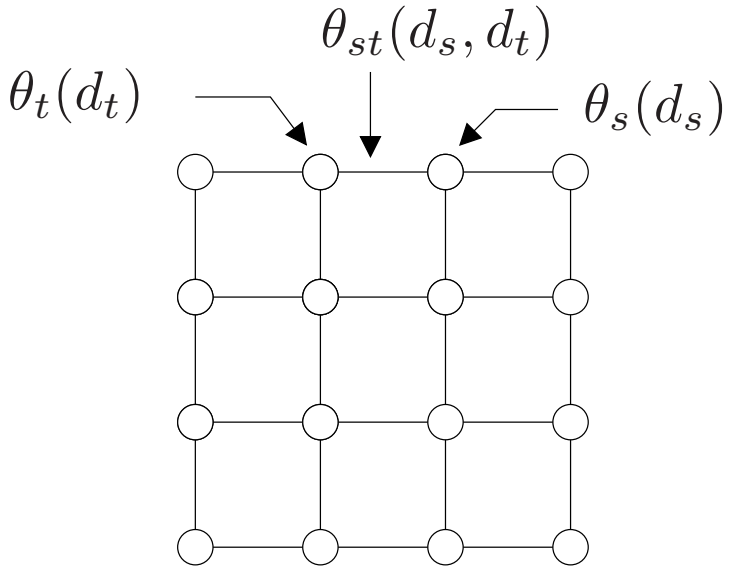
(c) Multiscale quadtree

- frequently wish to compute log likelihoods (e.g., for classification), or marginals/modes (e.g., for denoising, deblurring, de-convolution, coding)

- exact algorithms available for tree-structured models; approximate techniques (e.g., belief propagation and variants) required for more complex models

Example: Computer vision (II)

- disparity for stereo vision: estimate depth in scenes based on two (or more) images taken from different positions
- *global approaches*: disparity map based on optimization in an MRF



- grid-structured graph $G = (V, E)$
- $d_s \equiv$ disparity at grid position s
- $\theta_s(d_s) \equiv$ image data fidelity term
- $\theta_{st}(d_s, d_t) \equiv$ disparity coupling

- optimal disparity map $\widehat{\mathbf{d}}$ found by solving MAP estimation problem for this Markov random field
- computationally intractable (NP-hard) in general, but iterative message-passing algorithms (e.g., belief propagation) solve many practical instances

Stereo pairs: Dom St. Stephan, Passau

Source: http://www.usm.maine.edu/ rhodes/

Example: Graphical codes for communication

Goal: Achieve reliable communication over a noisy channel.

- wide variety of applications: satellite communication, sensor networks, computer memory, neural communication
- error-control codes based on careful addition of redundancy, with their fundamental limits determined by Shannon theory
- key implementational issues: *efficient* construction, encoding and decoding
- very active area of current research: *graphical codes* (e.g., turbo codes, low-density parity check codes) and iterative message-passing algorithms (belief propagation; max-product)

Graphical codes and decoding (continued)

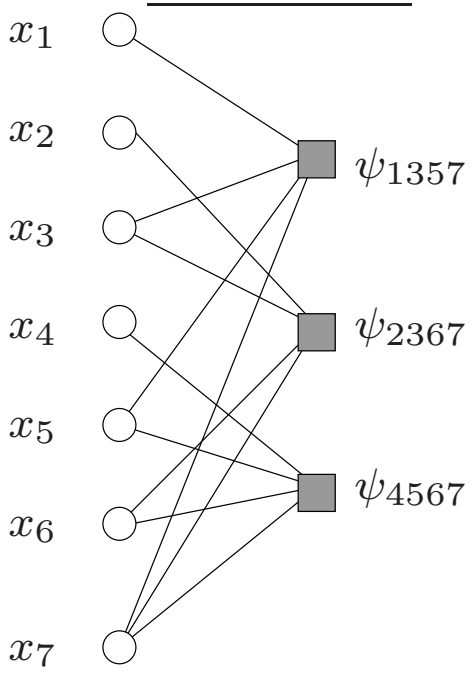
Parity check matrix

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Codeword: [0 1 0 1 0 1 0]

Non-codeword: [0 0 0 0 0 1 1]

Factor graph



- Decoding: requires finding maximum likelihood codeword:

$$\widehat{\mathbf{x}}_{ML} \;=\; \arg\max_{\mathbf{x}} p(\mathbf{y} \,|\, \mathbf{x}) \quad \text{s. t.} \quad H\mathbf{x} = 0 \;(\text{mod } 2).$$

- use of belief propagation as an approximate decoder has revolutionized the field of error-control coding

Outline

1. Background and set-up
   (a) Background on graphical models
   (b) Illustrative applications

2. Basics of graphical models
   (a) Classes of graphical models
   (b) Local factorization and Markov properties

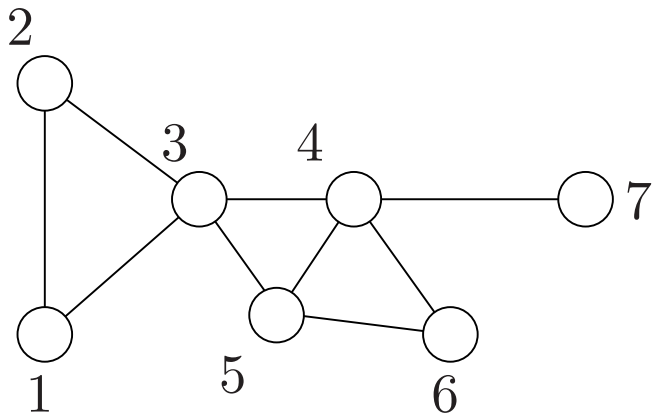
3. Exact message-passing on trees
   (a) Elimination algorithm
   (b) Sum-product and max-product on trees
   (c) Junction trees

4. Parameter estimation
   (a) Maximum likelihood
   (b) Proportional iterative fitting and related algorithsm
   (c) Expectation maximization

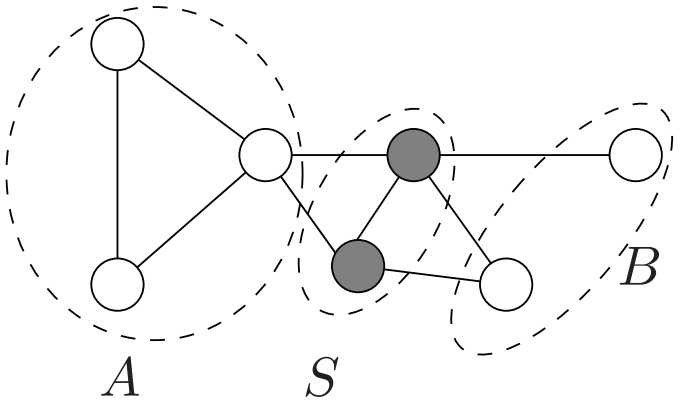
Undirected graphical models (Markov random fields)

Undirected graph combined with random vector $X = (X_1, \ldots, X_n)$

- given an undirected graph $G = (V, E)$, each node s has an associated random variable X_s
- for each subset $A \subseteq V$, define $X_A := \{X_s, s \in A\}$.



Maximal cliques (123), (345), (456), (47)



Vertex cutset S

- a clique $C \subseteq V$ is a subset of vertices all joined by edges
- a vertex cutset is a subset $S \subset V$ whose removal breaks the graph into two or more pieces

Factorization and Markov properties

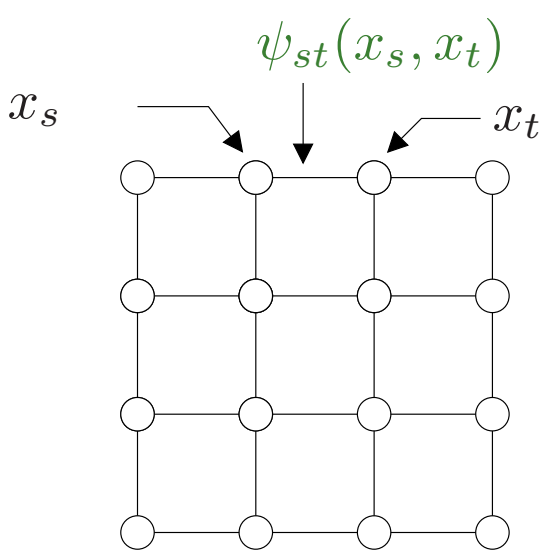
The graph G can be used to impose constraints on the random vector $X = X_V$ (or on the distribution p) in different ways.

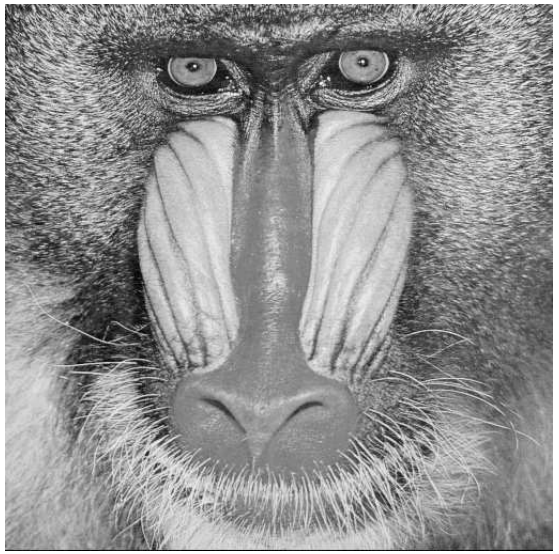
Markov property: X is *Markov w.r.t G* if X_A and X_B are conditionally indpt. given X_S whenever S separates A and B.

Factorization: The distribution p *factorizes according to G* if it can be expressed as a product over cliques:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \underbrace{\psi_C(x_C)}_{\text{compatibility function on clique } C}$$

Illustrative example: Ising/Potts model for images

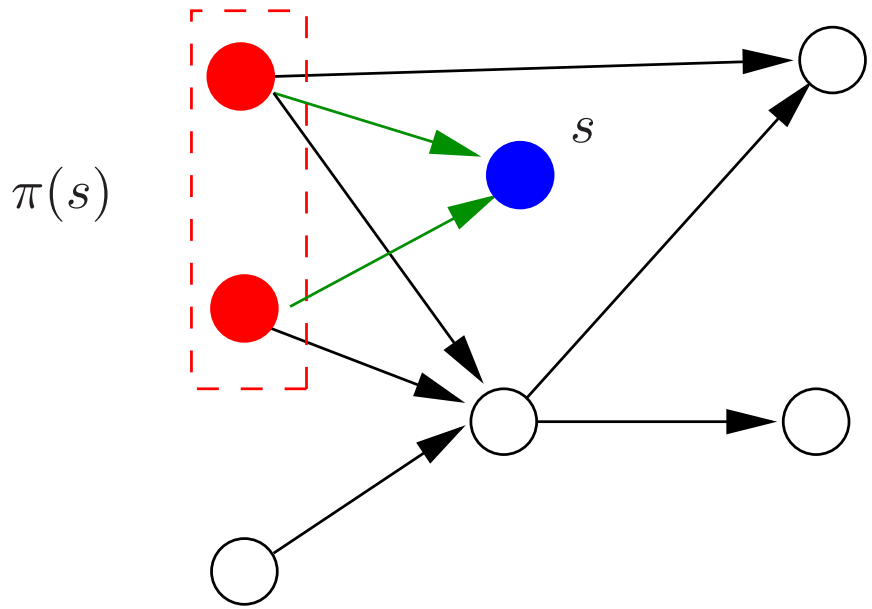




- discrete variables $X_s \in \{0, 1, \ldots, m-1\}$ (e.g., gray-scale levels)
- pairwise interaction $\psi_{st}(x_s, x_t) = \begin{bmatrix} a & b & b \\ b & a & b \\ b & b & a \end{bmatrix}$
- for example, setting $a > b$ imposes *smoothness* on adjacent pixels (i.e., $\{X_s = X_t\}$ more likely than $\{X_s \neq X_t\}$)

Directed graphical models

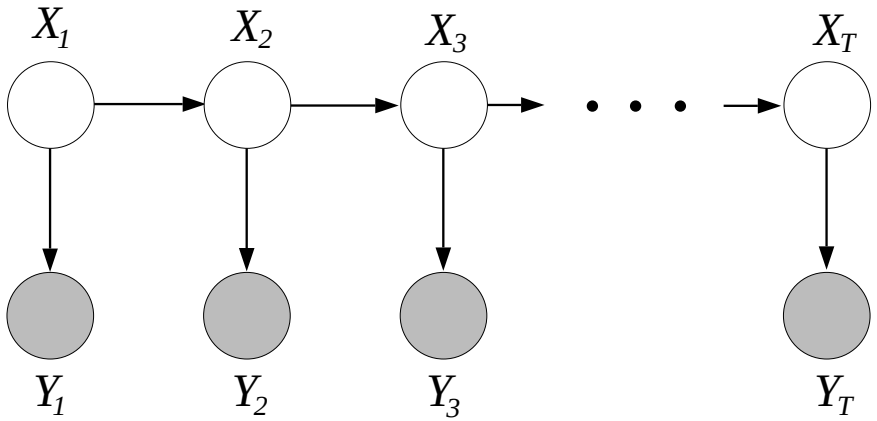
- factorization of probability distribution based on parent $\rightarrow$ child structure of a directed acyclic graph (DAG)



- denoting $\pi(s) = \{\text{set of parents of child } s\}$, we have:

$$p(x) \quad = \quad \prod_{s \in V} \underbrace{p\left(x_s \mid x_{\pi(s)}\right)}_{\text{parent-child conditional distributions}}$$

Illustrative example: Hidden Markov model (HMM)

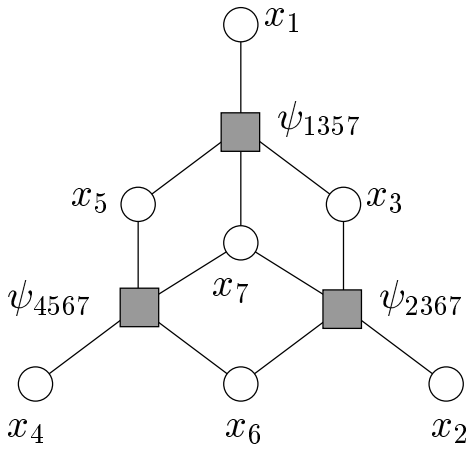


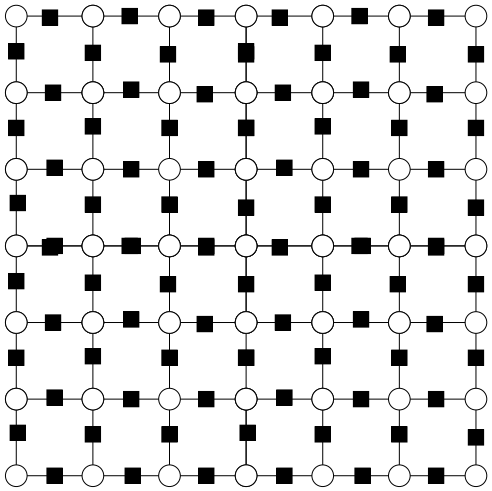
- hidden Markov chain $(X_1, X_2, \ldots, X_n)$ specified by conditional probability distributions $p(x_{t+1} \mid x_t)$
- noisy observations Y_t of X_t specified by conditional $p(y_t \mid x_t)$
- HMM can also be represented as an undirected model on the same graph with

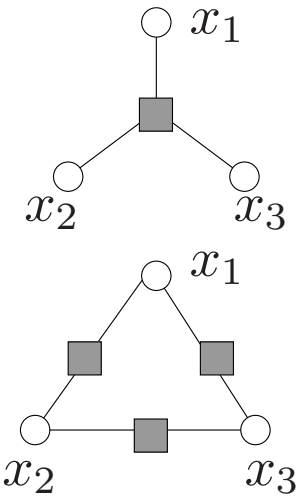
$$\begin{aligned} \psi_1(x_1) &= p(x_1) \\ \psi_{t,\,t+1}(x_t, x_{t+1}) &= p(x_{t+1} \mid x_t) \\ \psi_{tt}(x_t, y_t) &= p(y_t \mid x_t) \end{aligned}$$

Factor graph representations

- bipartite graphs in which
  - circular nodes (○) represent variables
  - square nodes (■) represent compatibility functions ψ_C







- factor graphs provide a finer-grained representation of factorization (e.g., 3-way interaction versus pairwise interactions)

- frequently used in communication theory

Representational equivalence: Factorization and Markov property

- both factorization and Markov properties are useful characterizations

Markov property: X is *Markov w.r.t G* if X_A and X_B are conditionally indpt. given X_S whenever S separates A and B.

Factorization: The distribution p *factorizes according to G* if it can be expressed as a product over cliques:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \underbrace{\psi_C(x_C)}_{\text{compatibility function on clique } C}$$

Theorem: (Hammersley-Clifford) For strictly positive $p(\cdot)$, the Markov property and the Factorization property are equivalent.

Outline

1. Background and set-up
   (a) Background on graphical models
   (b) Illustrative applications

2. Basics of graphical models
   (a) Classes of graphical models
   (b) Local factorization and Markov properties

3. Exact message-passing on trees
   (a) Elimination algorithm
   (b) Sum-product and max-product on trees
   (c) Junction trees

4. Parameter estimation
   (a) Maximum likelihood
   (b) Proportional iterative fitting and related algorithsm
   (c) Expectation maximization

Computational problems of interest

Given an undirected graphical model (possibly with observations $\mathbf{y}$):

$$p(\mathbf{x} \mid \mathbf{y}) \quad = \quad \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C) \prod_{s \in V} \psi_s(x_s; y_s)$$

Quantities of interest:

(a) the log normalization constant $\log Z$

(b) marginal distributions or other local statistics

(c) modes or most probable configurations

Relevant dimensions often grow rapidly in graph size $\Longrightarrow$ major computational challenges.

Example: Consider a naive approach to computing the normalization constant for binary random variables:

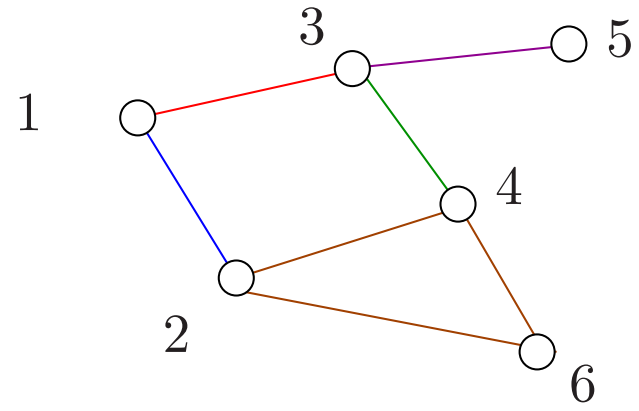
$$Z \; = \sum_{\mathbf{x} \in \{0,1\}^n} \prod_{C \in \mathcal{C}} \exp\left\{\psi_C(x_C)\right\}$$

Complexity scales exponentially as 2^n.

Elimination algorithm (I)

Suppose that we want to compute the marginal distribution $p(x_1)$:

$$\sum_{x_2,\ldots,x_6} \left[\psi_{12}(x_1,x_2)\psi_{13}(x_1,x_3)\psi_{34}(x_3,x_4)\psi_{35}(x_3,x_5)\psi_{246}(x_2,x_4,x_6)\right].$$

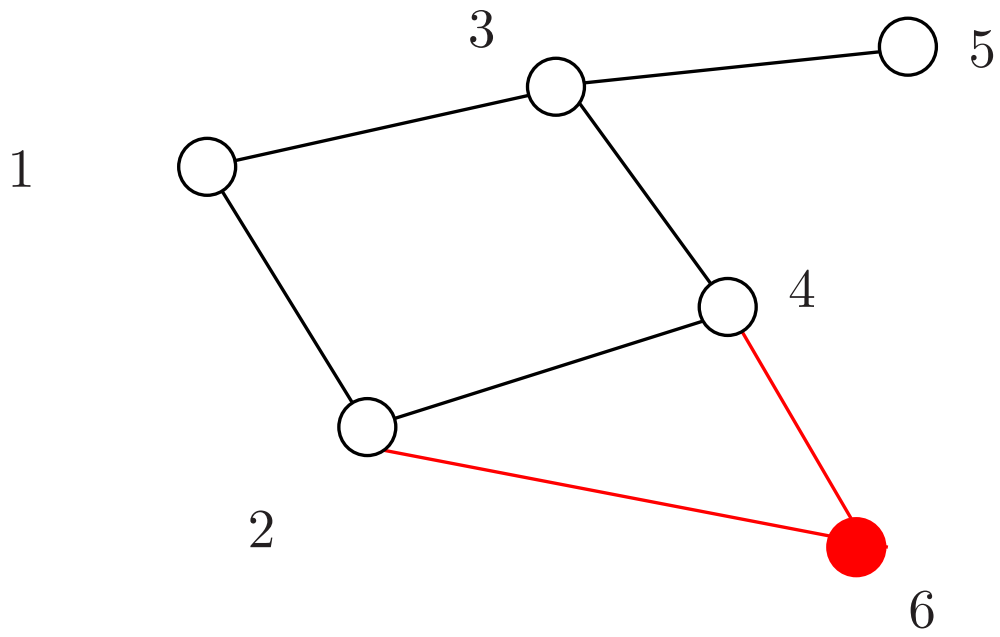


Exploit distributivity of sum and product operations:

$$p(x_1) \propto \sum_{x_2} \left[\psi_{12}(x_1,x_2) \sum_{x_3} \psi_{13}(x_1,x_3) \sum_{x_4} \psi_{34}(x_3,x_4) \times \right.$$
$$\left. \sum_{x_5} \psi_{35}(x_3,x_5) \sum_{x_6} \psi_{246}(x_2,x_4,x_6)\right].$$

Elimination algorithm (II;A)

Order of summation $\equiv$ order of removing nodes from graph.

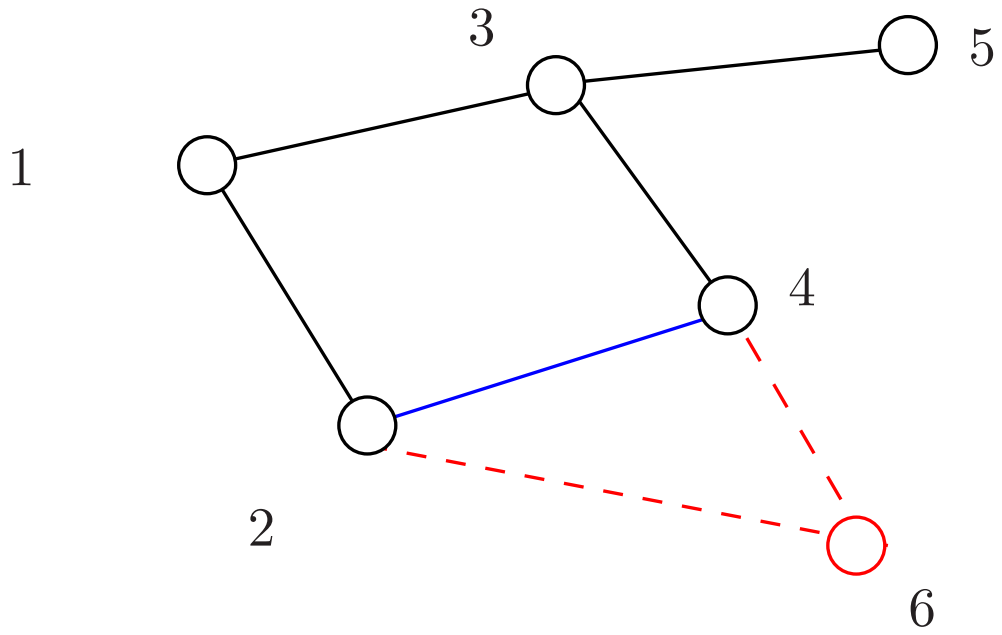


Summing over variable x_6 amounts to eliminating 6 and attached edges from the graph:

$$p(x_1) \propto \sum_{x_2} \Big[\psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) \sum_{x_4} \psi_{34}(x_3, x_4) \times \sum_{x_5} \psi_{35}(x_3, x_5) \Big\{ \sum_{x_6} \psi_{246}(x_2, x_4, x_6) \Big\} \Big].$$

Elimination algorithm (II;B)

Order of summation $\equiv$ order of removing nodes from graph.

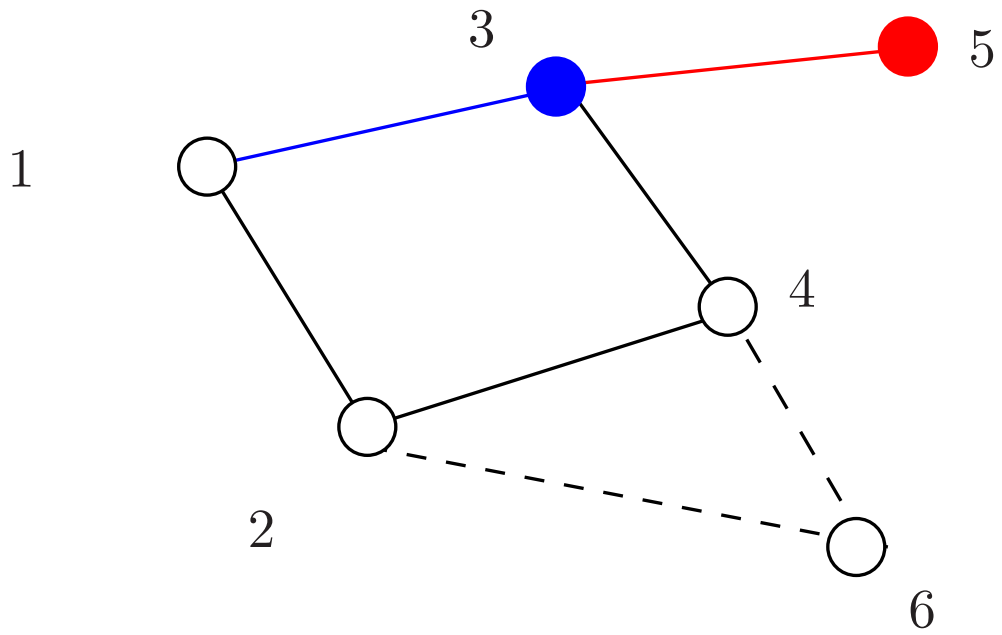


After eliminating x_6, left with a residual potential $\widetilde{\psi}_{24}$

$$p(x_1) \propto \sum_{x_2} \Big[\psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) \sum_{x_4} \psi_{34}(x_3, x_4) \times \sum_{x_5} \psi_{35}(x_3, x_5) \big\{\widetilde{\psi}_{24}(x_2, x_4)\big\}\Big].$$

Elimination algorithm (III)

Order of summation $\equiv$ order of removing nodes from graph.

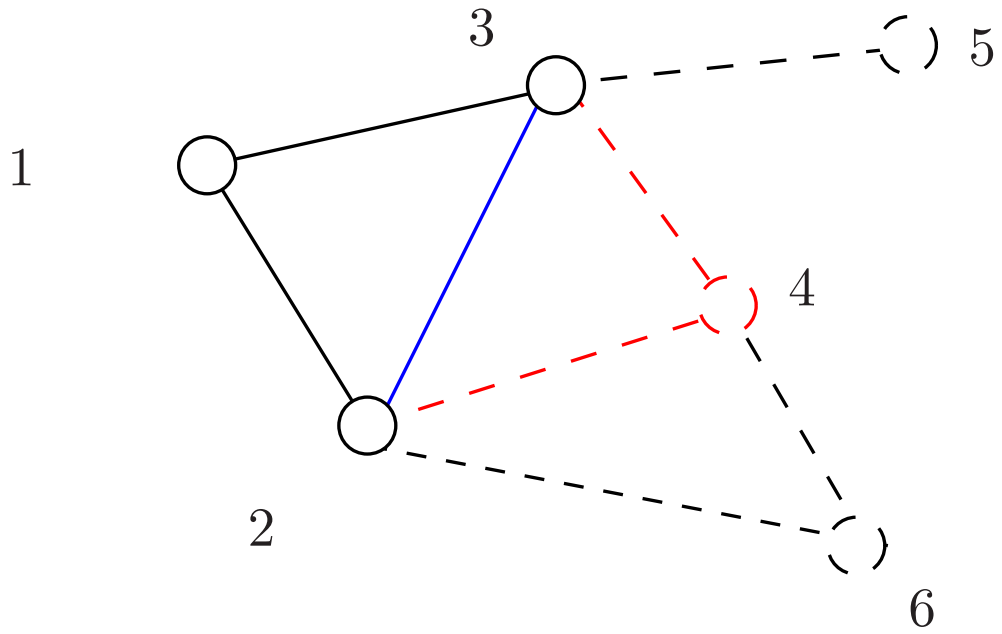


Similarly eliminating variable x_5 modifies $\psi_{13}(x_1, x_3)$:

$$p(x_1) \propto \sum_{x_2} \Big[\psi_{12}(x_1, x_2) \sum_{x_3} \psi_{13}(x_1, x_3) \sum_{x_4} \psi_{34}(x_3, x_4) \times \sum_{x_5} \psi_{35}(x_3, x_5) \widetilde{\psi}_{24}(x_2, x_4)\Big].$$

Elimination algorithm (IV)

Order of summation $\equiv$ order of removing nodes from graph.

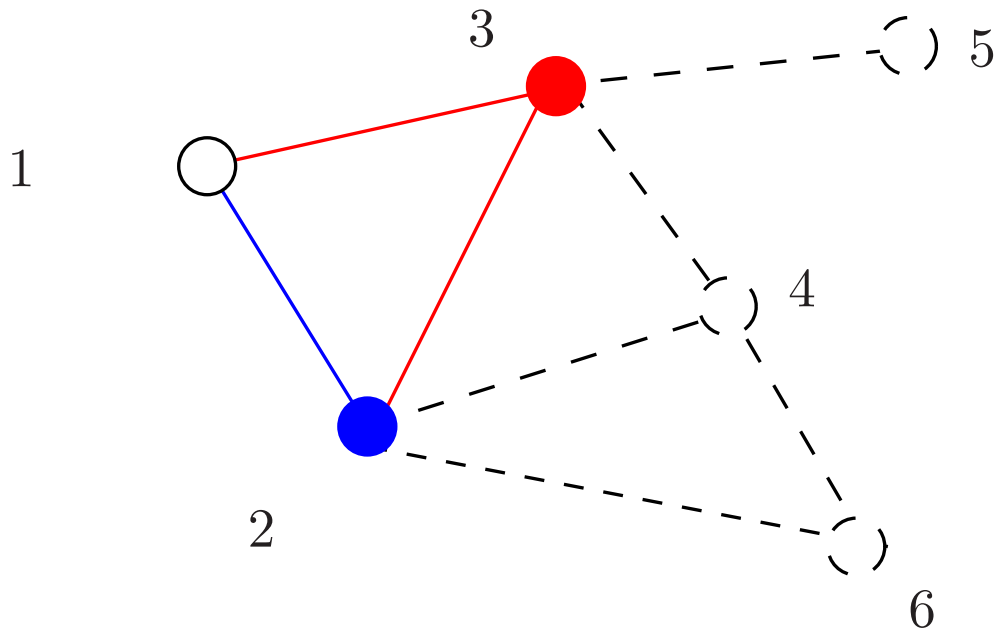


Eliminating variable x_4 leads to a new coupling term $\psi_{23}(x_2, x_3)$:

$$p(x_1) \quad \propto \quad \sum_{x_2} \psi_{12}(x_1, x_2) \sum_{x_3} \widetilde{\psi}_{13}(x_1, x_3) \underbrace{\sum_{x_4} \psi_{34}(x_3, x_4) \widetilde{\psi}_{24}(x_2, x_4)}_{\psi_{23}(x_2, x_3)}.$$

Elimination algorithm (V)

Order of summation $\equiv$ order of removing nodes from graph.



Finally summing/eliminating x_2 and x_3 yields the answer:

$$p(x_1) \quad \propto \quad \sum_{x_2} \psi_{12}(x_1, x_2) \Big[\sum_{x_3} \widetilde{\psi}_{13}(x_1, x_3) \psi_{13}(x_2, x_3) \Big].$$

Summary of elimination algorithm

- Exploits distributive law with sum and product to perform partial summations in a particular order.

- Graphical effect of summing over variable x_i:

  (a) eliminate node i and all its edges from the graph, and

  (b) join all neighbors $\{j \mid (i,j) \in E\}$ with a residual edge

- Computational complexity depends on the clique size of the residual graph that is created.

- Choice of elimination ordering

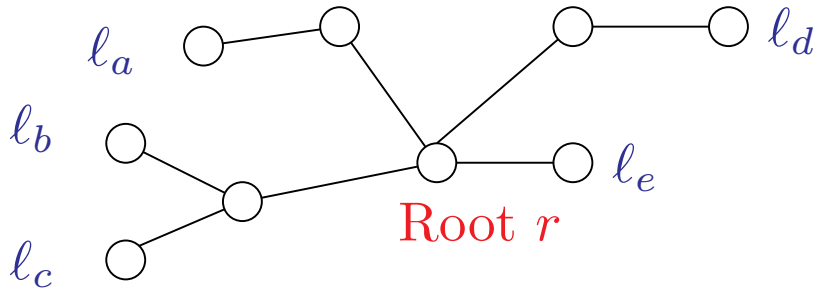
  (a) Desirable to choose ordering that leads to small residual graph.

  (b) Optimal choice of ordering is NP-hard.

Sum-product algorithm on (junction) trees

- sum-product generalizes many special purpose algorithms
  - transfer matrix method (statistical physics)
  - $\alpha - \beta$ algorithm, forward-backward algorithm (e.g., Rabiner, 1990)
  - Kalman filtering (Kalman, 1960; Kalman & Bucy, 1961)
  - peeling algorithms (Felsenstein, 1973)

- given an undirected tree (graph wtihout cycles) with root node, elimination $\equiv$ leaf-stripping



- suppose that we wanted to compute marginals $p(x_s)$ at *all nodes simultaneously*
  - running elimination algorithm n times (once for each node $s \in V$) fails to recycle calculations — very wasteful!

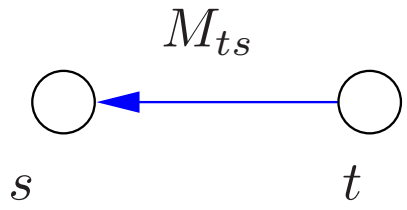
Sum-product on trees (I)

- sum-product algorithm provides an $O(n)$ algorithm for computing marginal at every tree node simultaneously

- consider the following parameterization on a tree:

$$p(\mathbf{x}) \quad = \quad \frac{1}{Z} \prod_{s \in V} \psi_s(x_s) \prod_{(s,t) \in E} \psi_{st}(x_s, x_t)$$

- simplest example: consider effect of eliminating x_t from edge (s, t)



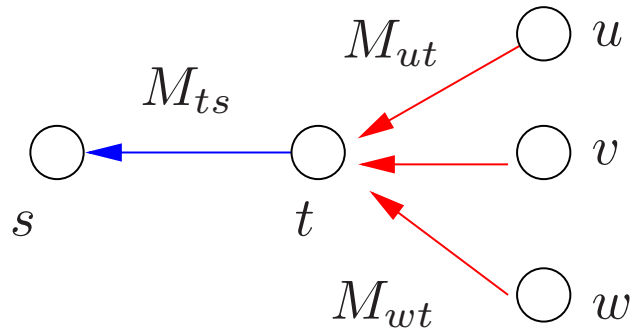
- effect of eliminating x_t can be represented as "message-passing":

$$p(x_1) \quad \propto \quad \psi_s(x_s) \sum_{x_t} \underbrace{[\psi_t(x_t)\psi_{st}(x_s, x_t)]}$$

Message $M_{ts}(x_s)$ from $t \to s$

Sum-product on trees (II)

- children of t (when eliminated) introduce other "messages" M_{ut}, M_{vt}, M_{wt}



- correspondence between elimination and message-passing:

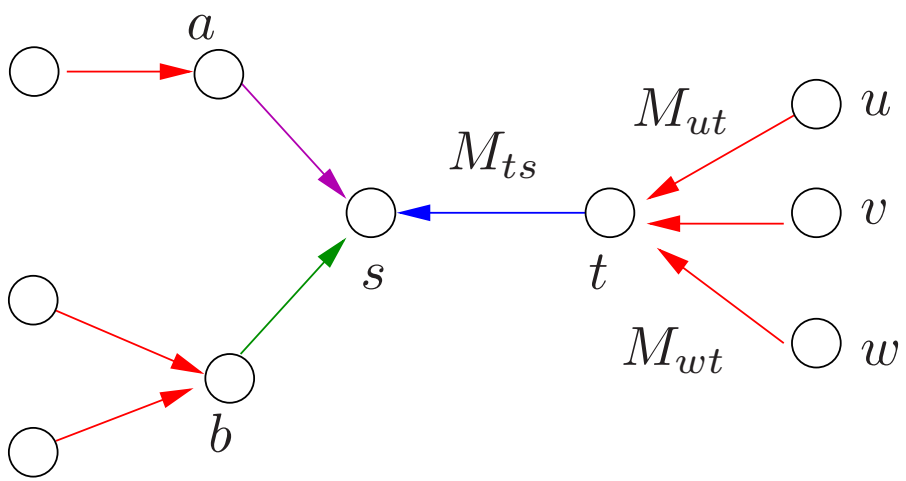
$$p(x_1) \quad \propto \quad \psi_s(x_s)\Big[\underbrace{\sum_{x_t} \psi_t(x_t)\psi_{st}(x_s,x_t)}_{\text{Message } t \to s} \underbrace{M_{ut}(x_t)M_{vt}(x_t)M_{wt}(x_t)}_{\text{Messages from children to } t}\Big]$$

- leads to the *message-update equation*:

$$M_{ts}(x_s) \quad \leftarrow \quad \sum_{x_t} \psi_t(x_t)\psi_{st}(x_s,x_t) \prod_{u \in N(t)\setminus s} M_{ut}(x_t)$$

Sum-product on trees (III)

- in general, node s has multiple neighbors (each the root of a subtree)

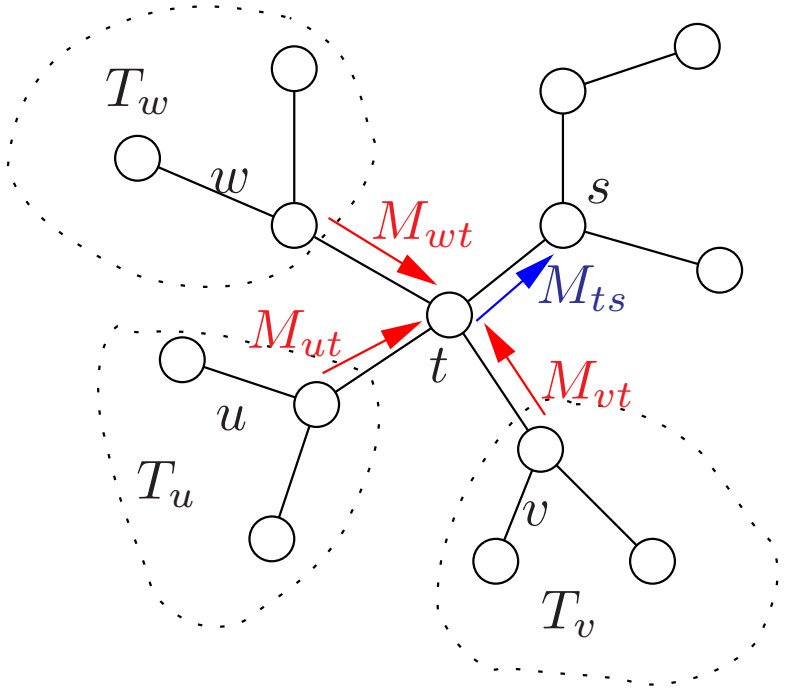


- marginal $p(x_s)$ can be computed from a product of incoming messages:

$$p(x_s) \quad \propto \quad \underbrace{\psi_s(x_s)}_{\text{Local evidence}} \qquad \underbrace{M_{ts}(x_s)\; M_{bs}(x_s)\;\; M_{as}(x_s)}_{\text{Contributions of neighboring subtrees}}$$

- sum-product updates are applied in parallel across entire tree

Summary: sum-product algorithm on a tree



$M_{ts} \equiv$ message from node t to s

$N(t) \equiv$ neighbors of node t

Sum-product: for marginals

(generalizes $\alpha - \beta$ algorithm; Kalman filter)

Proposition: On any tree, sum-product updates converge after a finite number of iterations (at most graph diameter), and yield exact marginal distributions.

Update: $$\mathbf{M_{ts}(x_s)} \leftarrow \sum_{x'_t \in \mathcal{X}_t} \left\{ \psi_{st}(x_s, x'_t)\, \psi_t(x'_t) \prod_{v \in N(t) \setminus s} \mathbf{M_{vt}(x_t)} \right\} .$$

Marginals: $$p(x_s) \propto \psi_t(x_t) \prod_{t \in N(s)} M_{ts}(x_s).$$

Max-product algorithm on trees (I)

- consider problem of computing a mode

$$\widehat{\mathbf{x}} \quad \in \quad \arg\max_{\mathbf{x}} p(\mathbf{x})$$

  of a distribution in graphical model form

- key observation: distributive law also applies to maximum and product operations, so global maximization can be broken down

- simple example: maximization of $p(\mathbf{x}) \propto \psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3)$ can be decomposed as

$$\max_{x_1} \Big\{ \Big[\max_{x_2} \psi_{12}(x_1, x_2) \Big] \quad \Big[\max_{x_3} \psi_{13}(x_1, x_3) \Big] \Big\}$$

- systematic procedure via max-product message-passing:
  - generalizes various special purpose methods (e.g., peeling techniques, Viterbi algorithm)
  - can be understood as *non-serial dynamic programming*

Max-product algorithm on trees (II)

- purpose of max-product: computing modes via max-marginals:

$$\widetilde{p}(x_1) \quad \propto \quad \max_{x_2', \ldots x_n'} p(x_1, x_2', x_3', \ldots, x_n')$$

partial maximization with x_1 held fixed

- max-product updates on a tree

$$M_{ts}(x_s) \quad \leftarrow \quad \max_{x_t' \in \mathcal{X}_t} \Big\{ \psi_{st}(x_s, x_t') \, \psi_t(x_t') \prod_{v \in N(t) \setminus s} M_{vt}(x_t) \Big\}.$$

Proposition: On any tree, max-product updates converge in a finite number of steps, and yield the max-marginals as:

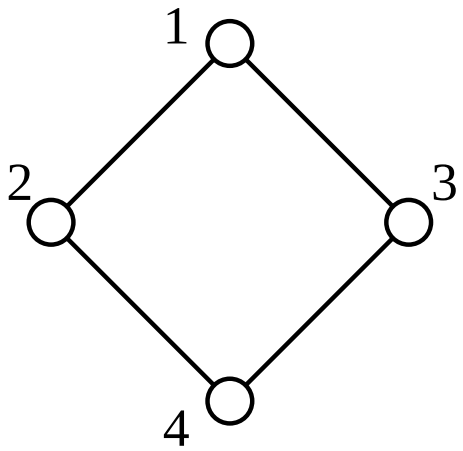
$$\widetilde{p}(x_s) \quad \propto \quad \psi_s(x_s) \prod_{t \in N(s)} M_{ts}(x_s).$$

From max-marginals, a mode can be determined by a "back-tracking" procedure on the tree.

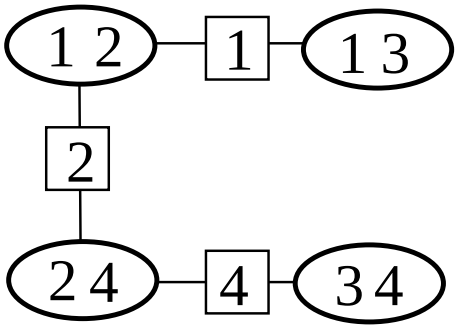
What to do for graphs with cycles?

Idea: Cluster nodes within cliques of graph with cycles to form a clique tree. Run a standard tree algorithm on this clique tree.

Caution: A naive approach will fail!



(a) Original graph



(b) Clique tree

Need to enforce consistency between the copy of x_3 in cluster $\{1, 3\}$ and that in $\{3, 4\}$.

Running intersection and junction trees

Definition: A clique tree satisfies the *running intersection property* if for any two clique nodes $\mathcal{C}_1$ and $\mathcal{C}_2$, all nodes on the unique path joining them contain the intersection $\mathcal{C}_1 \cap \mathcal{C}_2$.

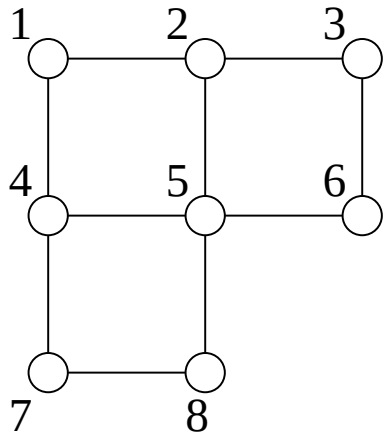
Key property: Running intersection ensures probabilistic consistency of calculations on the clique tree.

A clique tree with this property is known as a junction tree.

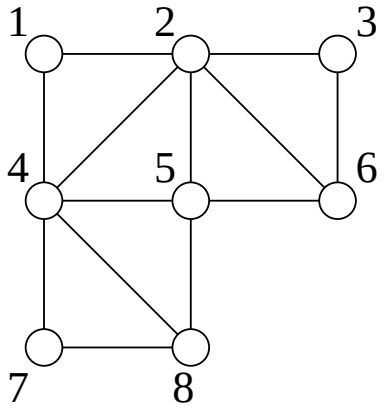
Question: What types of graphs have junction trees?

Junction trees and triangulated graphs

Definition: A graph is *triangulated* means that every cycle of length four or longer has a chord.



(a) Untriangulated



(b) Triangulated version

Proposition: A graph G has a junction tree if and only if it is triangulated. (Lauritzen, 1996)

Junction tree for exact inference

Algorithm: (Lauritzen & Spiegelhalter, 1988)

1. Given an undirected graph G, form a triangulated graph $\widetilde{G}$ by adding edges as necessary.
2. Form the clique graph (in which nodes are cliques of the triangulated graph).
3. Extract a junction tree (using a maximum weight spanning tree algorithm on weighted clique graph).
4. Run sum/max-product on the resulting junction tree.

Note: Separator sets are formed by the intersections of cliques adjacent in the junction tree.

Theoretical justification of junction tree

A. **Theorem:** For an undirected graph G, the following properties are equivalent:

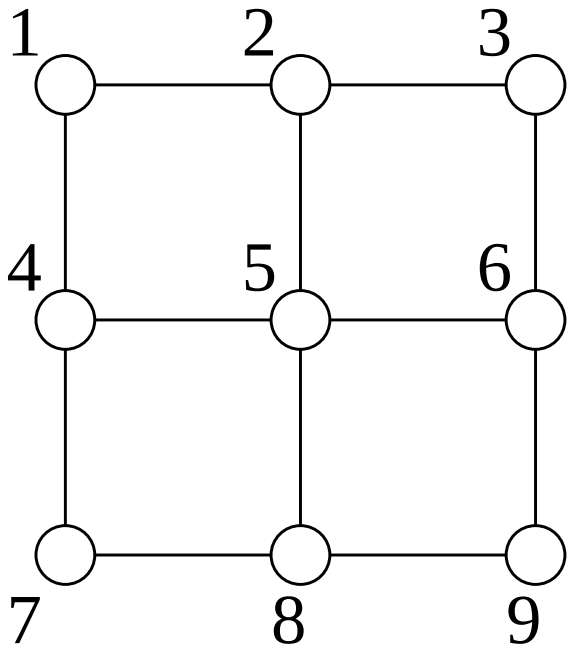
   (a) Graph G is triangulated.

   (b) The clique graph of G has a junction tree.

   (c) There is an elimination ordering for G that does not lead to any added edges.

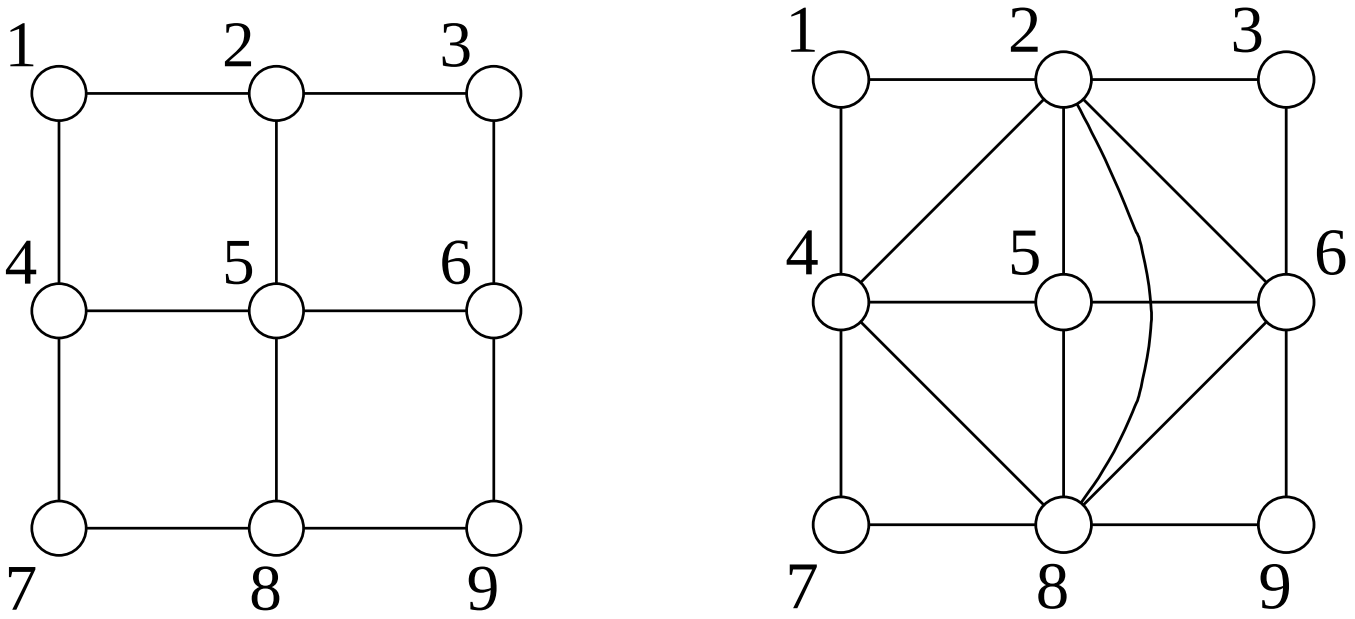
B. **Theorem:** Given a triangulated graph, weight the edges of the clique graph by the cardinality $|A \cap B|$ of the intersection of the adjacent cliques A and B. Then any maximum-weight spanning tree of the clique graph is a junction tree.

Illustration of junction tree (I)



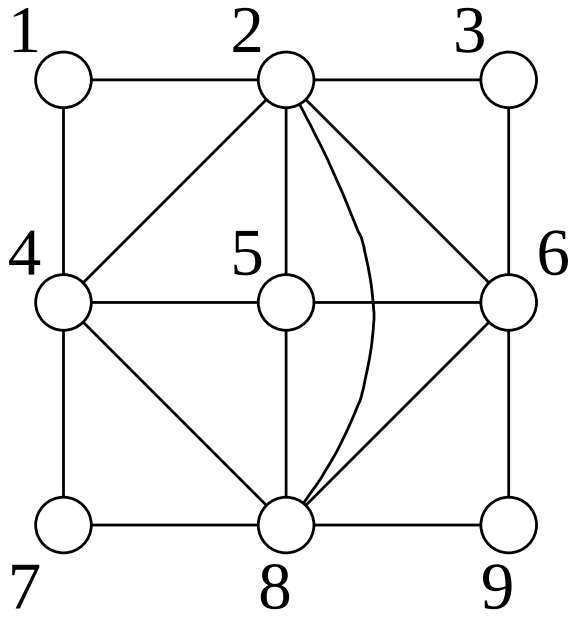
1. Begin with (original) untriangulated graph.

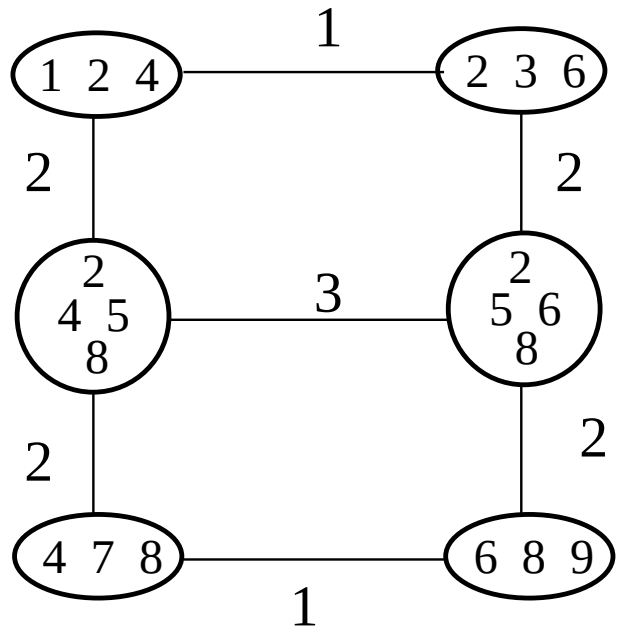
Illustration of junction tree (II)



2. Triangulate the graph by adding edges to remove chordless cycles.

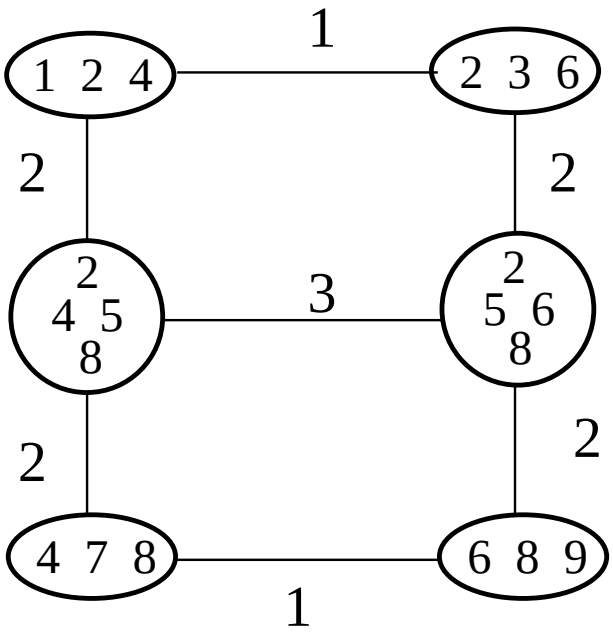
Illustration of junction tree (III)

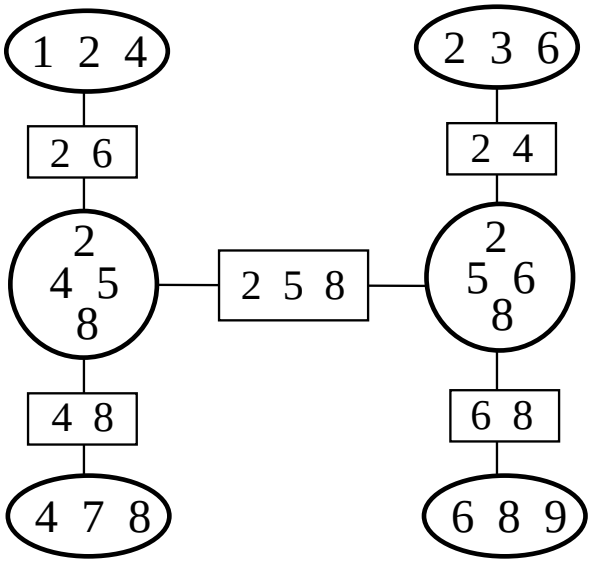




3. Form the clique graph associated with the triangulated graph. Place weights on the edges of the clique graph, corresponding to the cardinality of the intersections (separator sets).

Illustration of junction tree (IV)





4. Run a maximum weight spanning tree algorithm on the weight clique graph to find a junction tree. Run standard algorithms on the resulting tree.

Comments on junction tree algorithm

- treewidth of a graph $\equiv$ size of largest clique in junction tree
- complexity of running tree algorithms scales *exponentially* in size of largest clique
- junction tree depends critically on triangulation ordering (same as elimination ordering)
- choice of optimal ordering is NP-hard, but good heuristics exist
- junction tree formalism widely used, but limited to graphs of *bounded treewidth*

Junction tree representation

Junction tree representation guarantees that $p(\mathbf{x})$ can be factored as:

$$p(\mathbf{x}) = \frac{\prod_{\mathcal{C} \in \mathbf{C}_{\max}} p(\mathbf{x}_{\mathcal{C}})}{\prod_{\mathcal{S} \in \mathbf{C}_{\text{sep}}} p(\mathbf{x}_{\mathcal{S}})}$$

where

$\mathbf{C}_{\max}$ $\equiv$ set of all maximal cliques in *triangulated* graph $\widetilde{G}$

$\mathbf{C}_{\text{sep}}$ $\equiv$ set of all separator sets (intersections of adjacent cliques)

Special case for tree:

$$p(\mathbf{x}) = \prod_{s \in V} p(x_s) \prod_{(s,t) \in E} \frac{p(x_s, x_t)}{p(x_s)p(x_t)}$$

Outline

1. Background and set-up
   (a) Background on graphical models
   (b) Illustrative applications

2. Basics of graphical models
   (a) Classes of graphical models
   (b) Local factorization and Markov properties

3. Exact message-passing on trees
   (a) Elimination algorithm
   (b) Sum-product and max-product on trees
   (c) Junction trees

4. Parameter estimation
   (a) Maximum likelihood
   (b) Proportional iterative fitting and related algorithsm
   (c) Expectation maximization

Parameter estimation

- to this point, have assumed that model parameters (e.g., ψ_{123}) and graph structure were *known*

- in many applications, both are *unknown* and must be estimated on the basis of available data

Suppose that we are given independent and identically distributed samples $\mathbf{z}^1, \ldots, \mathbf{z}^N$ from unknown model $p(\mathbf{x}; \psi)$:

- **Parameter estimation:** Assuming knowledge of the graph structure, how to estimate the compatibility functions ψ?

- **Model selection:** How to estimate the graph structure (possibly in addition to ψ)?

Maximum likelihood (ML) for parameter estimation

- choose parameters ψ to maximize log likelihood of data

$$L(\psi; \mathbf{z}) \quad = \quad \frac{1}{N} \log p(\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(N)}; \psi) \quad \overset{(b)}{=} \quad \frac{1}{N} \sum_{i=1}^{N} \log p(\mathbf{z}^{(i)}; \psi)$$

  where equality (b) follows from independence assumption

- under mild regularity conditions, ML estimate $\widehat{\psi}^{ML} := \arg\max L(\psi; \mathbf{z})$ is asymptotically consistent

$$\widehat{\psi}^{ML} \xrightarrow{d} \psi^*$$

- for small sample sizes N, *regularized ML* frequently better behaved:

$$\widehat{\psi}^{RML} \quad \in \quad \arg\max_{\psi} \{L(\psi; \mathbf{z}) + \lambda \|\psi\|\}$$

  for some regularizing function $\|\cdot\|$.

ML optimality in fully observed models (I)

- convenient for studying optimality conditions: rewrite

$$\psi_C(x_C) = \quad \exp\left\{\sum_J \theta_{C;J} \mathbb{I}\left[x_C = J\right]\right\}$$

  where $\mathbb{I}\left[x_C = J\right]$ is an indicator function

- Example:

$$\psi_{st}(x_s, x_t) = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} = \begin{bmatrix} \exp(\theta_{00}) & \exp(\theta_{01}) \\ \exp(\theta_{10}) & \exp(\theta_{11}) \end{bmatrix}$$

- with this reformulation, log likelihood can be written in the form

$$L(\theta; \mathbf{z}) \quad = \quad \sum_C \theta_C \widehat{\mu}_C - \log Z(\theta)$$

  where $\widehat{\mu}_{C,J} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}\left[\mathbf{z}^{(i)} = J\right]$ are the empirical marginals

ML optimality in fully observed models (II)

- taking derivatives with respect to θ yields

$$\underbrace{\widehat{\mu}_{C,J}}_{\text{Empirical marginals}} \quad = \quad \frac{\partial}{\partial \theta_{C,J}} \log Z(\theta) \quad = \quad \underbrace{\mathbb{E}_\theta \left\{ \mathbb{I}\left[x_C = J\right] \right\}}_{\text{Model marginals}}$$

- ML optimality $\Longleftrightarrow$ empirical marginals are *matched* to model marginals

- one iterative algorithm for ML estimation: generate sequence of iterates $\{\theta^n\}$ by *naive gradient ascent*:

$$\theta^{n+1}_{C,J} \quad \leftarrow \quad \theta^n_{C,J} + \alpha \underbrace{\left[\widehat{\mu}_{C,J} - \mathbb{E}\{ \mathbb{I}\left[x_C = J\right] \} \right]}_{\text{current error}}$$

Iterative proportional fitting (IPF)

Alternative iterative algorithm for solving ML optimality equations:
(due to Darroch & Ratcliff, 1972)

1. Initialize ψ^0 to uniform functions (or equivalently, $\theta^0 = 0$).
2. Choose a clique C and update associated compatibility function:

$$\text{Scaling form:} \quad \psi_C^{(n+1)} = \psi_C^{(n)} \frac{\widehat{\mu}_C}{\mu_C(\psi^n)}$$

$$\text{Exponential form:} \quad \theta_C^{(n+1)} = \theta_C^{(n)} + [\log \widehat{\mu}_C - \log \mu_C(\theta^n)] .$$

   where $\mu_C(\psi^n) \equiv \mu_C(\theta^n)$ are current marginals predicted by model
3. Iterate updates until convergence.

Comments

- IPF $\equiv$ co-ordinate ascent on the log likelihood.
- special case of successive projection algorithms (Csiszar & Tusnady, 1984)

Parameter estimation in partially observed models

- many models allow only partial/noisy observations—say $p(\mathbf{y} \mid \mathbf{x})$—of the hidden quantities $\mathbf{x}$

- log likelihood now takes the form

$$L(\mathbf{y};\theta) \quad = \quad \frac{1}{N}\sum_{i=1}^{N} \log p(\mathbf{y}^{(i)};\theta) \;=\; \frac{1}{N}\sum_{i=1}^{N} \log \left\{ \sum_{\mathbf{z}} p(\mathbf{y}^{(i)} \mid \mathbf{z}) p(\mathbf{z};\theta) \right\}.$$

Data augmentation:

- suppose that we had observed the complete data $(\mathbf{y}^{(i)};\mathbf{z}^{(i)})$, and define complete log likelihood

$$C_{like}(\mathbf{z},\mathbf{y};\theta) \quad = \quad \frac{1}{N}\sum_{i=1}^{N} \log p(\mathbf{z}^{(i)},\mathbf{y}^{(i)};\theta).$$

- in this case, ML estimation would reduce to fully observed case
- **Strategy:** Make educated guesses at hidden data, and then average over the uncertainty.

Expectation-maximization (EM) algorithm (I)

- iterative updates on pair $\{\theta^n, q^n(\mathbf{z} \mid \mathbf{y})\}$ where
  - $\theta^n \equiv$ current estimate of the parameters
  - $q^n(\mathbf{z} \mid \mathbf{y}) \equiv$ current predictive distribution

EM Algorithm:

1. **E-step**: Compute expected complete log likelihood:

$$\mathcal{E}(\theta, \theta^n; q^n) \quad = \quad \sum_{\mathbf{z}} q^n(\mathbf{z} \mid \mathbf{y}; \theta^n) C_{like}(\mathbf{z}, \mathbf{y}; \theta)$$

   where $q^n(\mathbf{z} \mid \mathbf{y}; \theta^n)$ is conditional distribution under model $p(\mathbf{z}; \theta^n)$.

2. **M-step** Update parameter estimate by maximization

$$\theta^{n+1} \quad \leftarrow \quad \arg\max_{\theta} \mathcal{E}(\theta, \theta^n; q^n)$$

Expectation-maximization (EM) algorithm (II)

- alternative interpretation in lifted space (Csiszar & Tusnady, 1984)
- recall Kullback-Leibler divergence between distributions

$$D(r \,\|\, s) \quad = \quad \sum_{\mathbf{x}} r(\mathbf{x}) \log \frac{r(\mathbf{x})}{s(\mathbf{x})}$$

- KL divergence is one measure of distance between probability distributions
- link to EM: define an *auxiliary function*

$$\mathcal{A}(q, \theta) \quad = \quad D\Big(q(\mathbf{z} \,|\mathbf{y}) \widehat{p}(\mathbf{y}) \,\|\, p(\mathbf{z}, \mathbf{y}; \theta) \Big)$$

where $\widehat{p}(\mathbf{y})$ is the *empirical distribution*

Expectation-maximization (EM) algorithm (III)

- maximum likelihood (ML) equivalent to minimizing KL divergence

$$\underbrace{D\big(\widehat{p}(\mathbf{y}) \,\|\, p(\mathbf{y};\theta)\big)}_{\text{KL emp. vs fit}} \;=\; -H(\widehat{p}) \underbrace{-\sum_{\mathbf{y}} \widehat{p}(\mathbf{y}) \log p(\mathbf{y};\theta)}_{\text{negative log likelihood}}$$

Lemma: Auxiliary function is an upper bound on desired KL divergence:

$$D\big(\widehat{p}(\mathbf{y}) \,\|\, p(y;\theta)\big) \;\leq\; \mathcal{A}(q,\theta) \;=\; D\Big(q(\mathbf{z}\,|\mathbf{y})\widehat{p}(\mathbf{y}) \,\|\, p(\mathbf{z},\mathbf{y};\theta)\Big)$$

for all choices of q, with equality for $q(\mathbf{z}\,|\,\mathbf{y}) = p(\mathbf{z}\,|\mathbf{y};\theta)$.

EM algorithm is simply co-ordinate descent on this auxiliary function:

1. **E-step:** $q(\mathbf{z}\,|\mathbf{y},\theta^n) = \arg\min_q \mathcal{A}(q,\theta^n)$
2. **M-step:** $\theta^{n+1} = \arg\min_\theta \mathcal{A}(q^n,\theta)$