Probabilistic Graphical Models

Lecture 2: Graphical Models. Belief Propagation

Volkan Cevher, Matthias Seeger
Ecole Polytechnique Fédérale de Lausanne

30/9/2011

Outline

1. Graphical Models
2. Belief Propagation

Literature

Excellent book about graphical models and belief propagation, written by one of the pioneers in these topics:

- Pearl, J.
  Probabilistic Reasoning in Intelligent Systems (1990)

The Need to Factorize

Variables $x_1, x_2, \ldots, x_n$

$$P(x_1) = \sum_{x_2} \cdots \sum_{x_n} P(x_1, x_2, \ldots, x_n)$$

Marginalization: Exponential time
Storage: Exponential space $\Rightarrow$ Need factorization

- Independence?
  But probabilistic modelling is about dependencies!

The Need to Factorize

Variables $x_1, x_2, \ldots, x_n$

$$P(x_1) = \sum_{x_2} \cdots \sum_{x_n} P(x_1, x_2, \ldots, x_n)$$

Marginalization: Exponential time
Storage: Exponential space $\Rightarrow$ Need factorization

- Independence?
  But probabilistic modelling is about dependencies!
- Conditional independence
  Dependencies may have simple structure

Towards Bayesian Networks

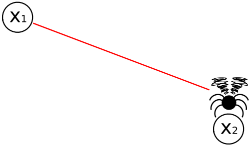
Tracking a fly

- Path pretty random

Towards Bayesian Networks

Tracking a fly

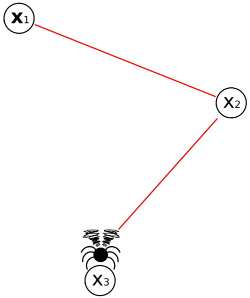
- Path pretty random



Towards Bayesian Networks

Tracking a fly

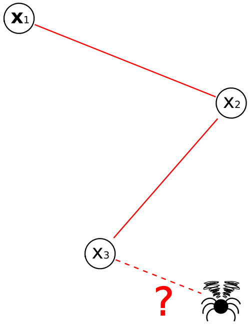
- Path pretty random
- Positions not independent



Towards Bayesian Networks

Tracking a fly

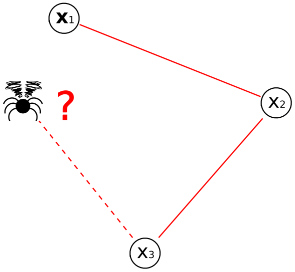
- Path pretty random
- Positions not independent



Towards Bayesian Networks

Tracking a fly

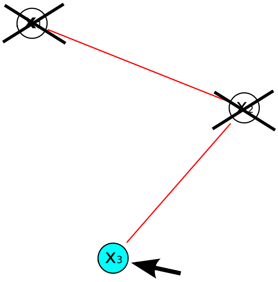
- Path pretty random
- Positions not independent



Towards Bayesian Networks

Tracking a fly

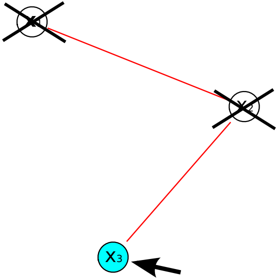
- Path pretty random
- Positions not independent
- But conditionally independent (Markovian)



Towards Bayesian Networks

Tracking a fly

- Path pretty random
- Positions not independent
- But conditionally independent (Markovian)



Remember

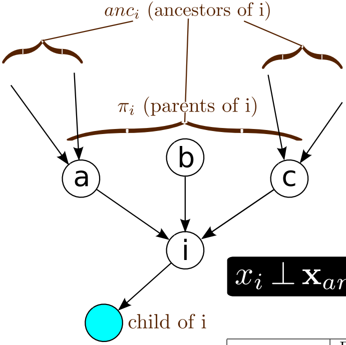
$$P(x_1, \ldots, x_n) = P(x_1)P(x_2|x_1)\ldots P(x_n|x_{n-1}, \ldots, x_1)\ ?$$

Here: $P(x_n|x_{n-1}, \ldots, x_1) = P(x_n|x_{n-1}) \Rightarrow$ Linear storage
Causal factorization $\Rightarrow$ Bayesian networks

Bayesian Networks (Directed Graphical Models)

Causal factorization:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | \boldsymbol{x}_{\pi_i})$$



$$x_i \perp \mathbf{x}_{anc_i} \mid \mathbf{x}_{\pi_i}$$

CPT:

x_a	x_b	x_c	$P(x_i = 1 \vert \mathbf{x}_{abc})$
0	0	0	0.25
0	0	1	0.1
			...
1	1	1	1

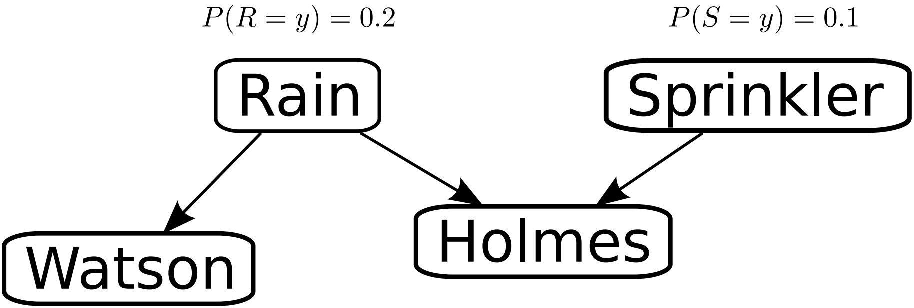
Bayesian network
(aka directed graphical model, aka causal network):

- Graphical representation of ancestry [DAG]
- $P(x_i|\boldsymbol{x}_{\pi_i})$: Conditional probability table (CPT)

Conditional Independence

$$\mathcal{A} \perp \mathcal{B} \,|\, \mathcal{C} \;\Leftrightarrow\; P(\mathcal{A}, \mathcal{B} \,|\, \mathcal{C}) = P(\mathcal{A} \,|\, \mathcal{C}) P(\mathcal{B} \,|\, \mathcal{C}) \;\Leftrightarrow\; P(\mathcal{A} \,|\, \mathcal{C}, \mathcal{B}) = P(\mathcal{A} \,|\, \mathcal{C})$$

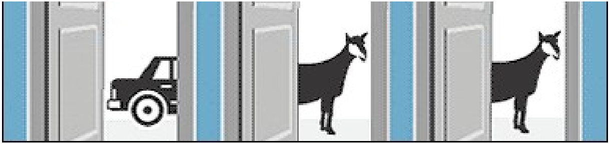
Did It Rain Tonight?



R	$P(W = y \mid R)$
y	1
n	0.2

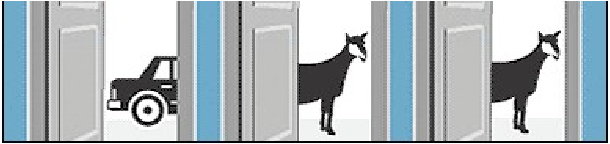
R	S	$P(H = y \mid R, S)$
y	y	1
y	n	1
n	y	0.9
n	n	0.01

Monty Hall Problem



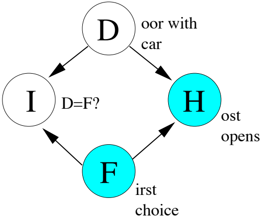
- Let's make a deal!
  - Door with car (hidden)
  - First choice of yours (remains closed)
  - Host opens door with goat, $H \neq F, D$
  - Do you switch?

Monty Hall Problem



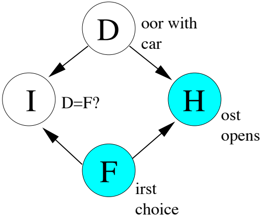
- Let's make a deal!
  - Door with car (hidden)
  - First choice of yours (remains closed)
  - Host opens door with goat, $H \neq F, D$
  - Do you switch?
- "Intuition": Fifty-fifty.
  F, H give no information. He would be stupid, wouldn't he?

Winning with Bayes (I)



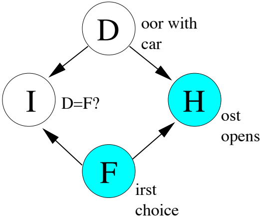
- Intuition "*H* does not tell anything" correct in principle. But about what?
- Add latent $I = \mathrm{I}_{\{D=F\}} = \mathrm{I}_{\{\text{first choice correct}\}}$

Winning with Bayes (I)



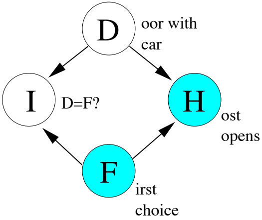
- Intuition "H does not tell anything" correct in principle. But about what?
- Add latent $I = \mathrm{I}_{\{D=F\}} = \mathrm{I}_{\{\text{first choice correct}\}}$
- Gut feeling: F, H no information about I. "He will not tell me whether I am correct". $P(I|F,H) = P(I)$. Will use Bayes to see that.

Winning with Bayes (I)



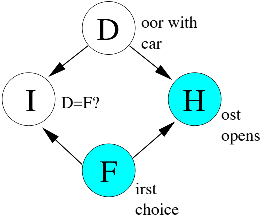
- Intuition "H does not tell anything" correct in principle. But about what?
- Add latent $I = \mathrm{I}_{\{D=F\}} = \mathrm{I}_{\{\text{first choice correct}\}}$
- Gut feeling: F, H no information about I. "He will not tell me whether I am correct". $P(I|F,H) = P(I)$. Will use Bayes to see that.
- OK, but $P(\text{Switch wins}) = P(I = 0|F,H) = P(I = 0) = 2/3$!

Winning with Bayes (I)



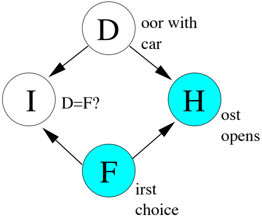
- Intuition "H does not tell anything" correct in principle. But about what?
- Add latent $I = \mathrm{I}_{\{D=F\}} = \mathrm{I}_{\{\text{first choice correct}\}}$
- Gut feeling: F, H no information about I.
  "He will not tell me whether I am correct".
  $P(I|F,H) = P(I)$.
  Will use Bayes to see that.
- OK, but $P(\text{Switch wins}) = P(I = 0|F,H) = P(I = 0) = 2/3$!
- Bayes makes you switch and double your chance of winning!

Winning with Bayes (II)



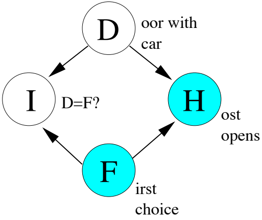
- To show: $P(I|H,F) = P(I)$.
- $P(I|F) = P(I)$,
  because D, F independent.

Winning with Bayes (II)



- To show: $P(I|H,F) = P(I)$.
- $P(I|F) = P(I)$,
  because D, F independent.
- $P(I|H,F) = P(I|F) \Leftrightarrow I \perp H|F \Leftrightarrow H \perp I|F$
  $\Leftrightarrow P(H|I,F) = P(H|F)$
  (independence is symmetric)

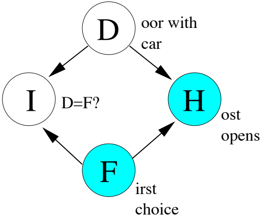
Winning with Bayes (II)



- To show: $P(I|H,F) = P(I)$.
- $P(I|F) = P(I)$,
  because D, F independent.
- $P(I|H,F) = P(I|F) \Leftrightarrow I \perp H | F \Leftrightarrow H \perp I | F$
  $\Leftrightarrow P(H|I,F) = P(H|F)$
  (independence is symmetric)

- $P(H|F, I=1) = (1/2)\mathrm{I}_{\{H \neq F\}}$
  If $F = D$, host picks random goat

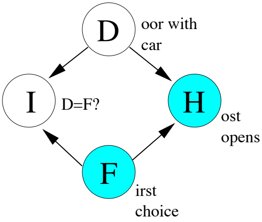
Winning with Bayes (II)



- To show: $P(I|H,F) = P(I)$.
- $P(I|F) = P(I)$,
  because D, F independent.
- $P(I|H,F) = P(I|F) \Leftrightarrow I \perp H | F \Leftrightarrow H \perp I | F$
  $\Leftrightarrow P(H|I,F) = P(H|F)$
  (independence is symmetric)

- $P(H|F, I=1) = (1/2)\mathrm{I}_{\{H \neq F\}}$
  If $F = D$, host picks random goat
- $P(H|F, I=0) = (1/2)\mathrm{I}_{\{H \neq F\}}$
  D, F independent, and $H \neq D, F$

Winning with Bayes (II)



- To show: $P(I|H,F) = P(I)$.
- $P(I|F) = P(I)$,
  because D, F independent.
- $P(I|H,F) = P(I|F) \Leftrightarrow I \perp H | F \Leftrightarrow H \perp I | F$
  $\Leftrightarrow P(H|I,F) = P(H|F)$
  (independence is symmetric)
- $P(H|F, I = 1) = (1/2)\mathrm{I}_{\{H \neq F\}}$
  If $F = D$, host picks random goat
- $P(H|F, I = 0) = (1/2)\mathrm{I}_{\{H \neq F\}}$
  D, F independent, and $H \neq D, F$

Working with Graphical Models

- Intermediate between lots of headscratching and doing all sums
- Powerful division of inference in manageable, local steps

Why Graphical Models?

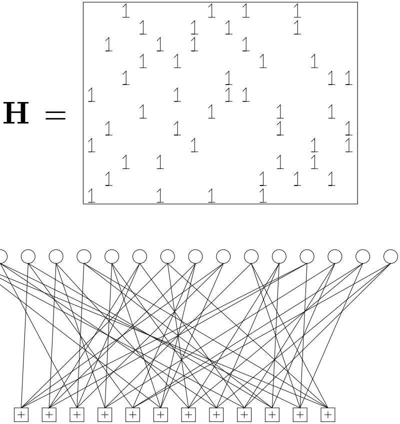
1. Easy way of communicating ideas about dependencies, models
2. Precise semantics: Conditional independence constraints on distributions. Efficient algorithms for testing these
3. Lead to large savings in computations (belief propagation)

Graphical Models in Practice

Dependency structures, and efficient ways to propagate information or constraints, are fundamental.

Coding / Information Theory

- LDPC codes and BP decoding revolutionized this field (resurrection of Gallager codes)
- Used from deep space communication (Mars rovers) over satellite transmission to CD players / hard drives



Courtesy MacKay: Information Theory . . . (2003)

Graphical Models in Practice

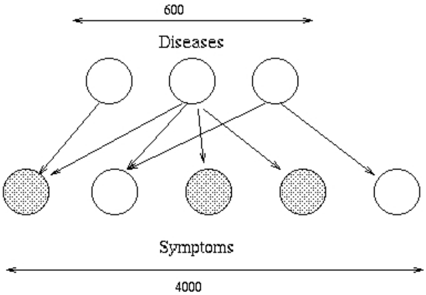
Dependency structures, and efficient ways to propagate information or constraints, are fundamental.

Expert systems done right

- QMR-DT: Invert causal network for helping medical diagnoses
- Hugin: Advanced decision support (Lauritzen)

  http://www.hugin.com/
- Promedas: Medical diagnostic advisory system (SNN Nimegen)

  http://www.promedas.nl/



Graphical Models in Practice

Dependency structures, and efficient ways to propagate information or constraints, are fundamental.

Computer Vision: Markov Random Fields

- Denoising, super-resolution, restoration (early work by Besag)
- Depth / reconstruction from stereo, matching, correspondences
- Segmentation, matting, blending, stitching, impainting, . . .

Courtesy MSR

Conditional Independence Semantics

- Graphical model formally equivalent to long (finite) list of conditional independence constraints:
  $\boldsymbol{x}_{A_1} \perp \boldsymbol{x}_{B_1} \mid \boldsymbol{x}_{C_1}, \boldsymbol{x}_{A_2} \perp \boldsymbol{x}_{B_2} \mid \boldsymbol{x}_{C_2}, \dots$ Which do you prefer?

Conditional Independence Semantics

- Graphical model formally equivalent to long (finite) list of conditional independence constraints:
  $\boldsymbol{x}_{A_1} \perp \boldsymbol{x}_{B_1} \mid \boldsymbol{x}_{C_1}, \boldsymbol{x}_{A_2} \perp \boldsymbol{x}_{B_2} \mid \boldsymbol{x}_{C_2}, \dots$ Which do you prefer?
- Graphs not just simpler for us:
  Linear-time algorithm to test such constraints (Bayes ball)

Conditional Independence Semantics

- Graphical model formally equivalent to long (finite) list of conditional independence constraints:
  $\boldsymbol{x}_{A_1} \perp \boldsymbol{x}_{B_1} \mid \boldsymbol{x}_{C_1}, \boldsymbol{x}_{A_2} \perp \boldsymbol{x}_{B_2} \mid \boldsymbol{x}_{C_2}, \dots$ Which do you prefer?
- Graphs not just simpler for us:
  Linear-time algorithm to test such constraints (Bayes ball)
- Distribution consistent with graph iff all CI constraints are met.
  $P(x_1)P(x_2)\dots P(x_n)$: Consistent with all graphs

Conditional Independence Semantics

- Graphical model formally equivalent to long (finite) list of conditional independence constraints:
  $\boldsymbol{x}_{A_1} \perp \boldsymbol{x}_{B_1} \mid \boldsymbol{x}_{C_1}, \boldsymbol{x}_{A_2} \perp \boldsymbol{x}_{B_2} \mid \boldsymbol{x}_{C_2}, \dots$ Which do you prefer?
- Graphs not just simpler for us:
  Linear-time algorithm to test such constraints (Bayes ball)
- Distribution consistent with graph iff all CI constraints are met.
  $P(x_1)P(x_2)\dots P(x_n)$: Consistent with all graphs
- How do I see whether $\boldsymbol{x}_A \perp \boldsymbol{x}_B \mid \boldsymbol{x}_C$ from the graph?
  Graph separation: If paths $A \leftrightarrow B$ blocked by C

Conditional Independence Semantics

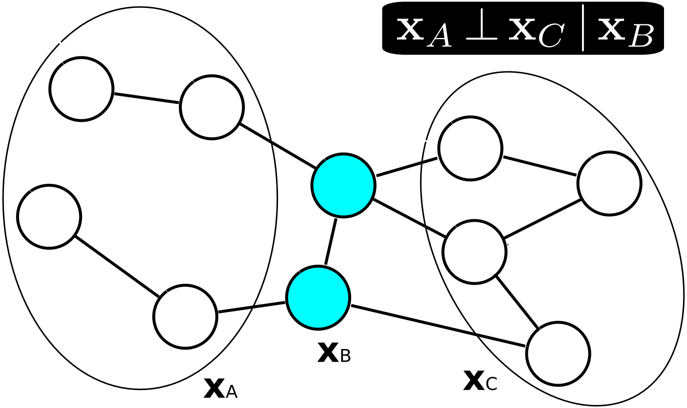
- Graphical model formally equivalent to long (finite) list of conditional independence constraints:
  $\boldsymbol{x}_{A_1} \perp \boldsymbol{x}_{B_1} \,|\, \boldsymbol{x}_{C_1},\ \boldsymbol{x}_{A_2} \perp \boldsymbol{x}_{B_2} \,|\, \boldsymbol{x}_{C_2}, \dots$ Which do you prefer?
- Graphs not just simpler for us:
  Linear-time algorithm to test such constraints (Bayes ball)
- Distribution consistent with graph iff all CI constraints are met.
  $P(x_1)P(x_2)\dots P(x_n)$: Consistent with all graphs
- How do I see whether $\boldsymbol{x}_A \perp \boldsymbol{x}_B \,|\, \boldsymbol{x}_C$ from the graph?
  Graph separation: If paths $A \leftrightarrow B$ blocked by C
- For Bayesian networks (directed graphical models): d-separation.
  $\Rightarrow$ You'll find out in the exercises!

Undirected Graphical Models (Markov Random Fields)

- Bayesian Networks: Describe CIs with directed graphs (DAGs)
  Markov Random Fields: Describe CIs with undirected graphs

Undirected Graphical Models (Markov Random Fields)

- Bayesian Networks: Describe CIs with directed graphs (DAGs)
  Markov Random Fields: Describe CIs with undirected graphs
- CI semantics of undirected models: Really just graph separation



Undirected Graphical Models (II)

- Why two frameworks?
  - Each can capture setups the other cannot
  - More important: In practice, some problems are much easier to parameterize (therefore: to learn) as MRFs, others much easier as Bayes nets

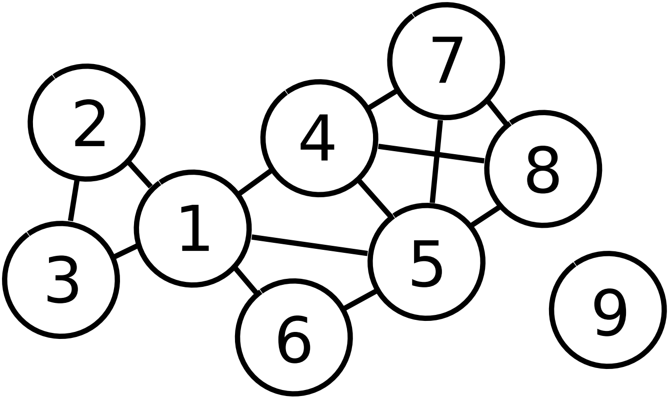
Undirected Graphical Models (II)

- Why two frameworks?
  - Each can capture setups the other cannot
  - More important: In practice, some problems are much easier to parameterize (therefore: to learn) as MRFs, others much easier as Bayes nets
- How do distributions P for MRF graph $\mathcal{G}$ look like? Hammersley / Clifford:
  - Maximal cliques (completely connected parts) C_j of $\mathcal{G}$
  - $P(\boldsymbol{x})$ consistent with MRF $G \Leftrightarrow$

$$P(\boldsymbol{x}) = Z^{-1} \prod_j \Phi_j(\boldsymbol{x}_{C_j}), \quad Z := \sum_{\boldsymbol{x}} \prod_j \Phi_j(\boldsymbol{x}_{C_j})$$

    with potentials $\Phi_j(\boldsymbol{x}_{C_j}) \geq 0$. Z: Partition function.
  - Potentials need not normalize to 1

Undirected Graphical Models (III)



$$P(\mathbf{x}) = Z^{-1}\phi_1(\mathbf{x}_{123})\phi_2(\mathbf{x}_{145})\phi_3(\mathbf{x}_{156})$$
$$\phi_4(\mathbf{x}_{4578})\phi_5(x_9)$$

Directed vs. Undirected

- Sampling $\boldsymbol{x} \sim P(\boldsymbol{x})$:
  Always simple from Bayes net. Can be very hard for an MRF

Directed vs. Undirected

- Sampling $\boldsymbol{x} \sim P(\boldsymbol{x})$:
  Always simple from Bayes net. Can be very hard for an MRF
- Implicit, symmetrical knowledge? Little idea about causal links (pixels of image, correspondences)? MRFs more useful then

Directed vs. Undirected

- Sampling $\boldsymbol{x} \sim P(\boldsymbol{x})$:
  Always simple from Bayes net. Can be very hard for an MRF
- Implicit, symmetrical knowledge? Little idea about causal links (pixels of image, correspondences)? MRFs more useful then
- Bottomline: Usually, one or the other is much more suitable. Better know well about both!

Towards Efficient Marginalization

- With sufficient Markovian CI constraints (directed or undirected):

$$P(x_1, \dots, x_n) \propto \prod_j \Phi_j(\boldsymbol{x}_{N_j}), \quad |N_j| \ll n$$

Can store that. But what about computation?

Towards Efficient Marginalization

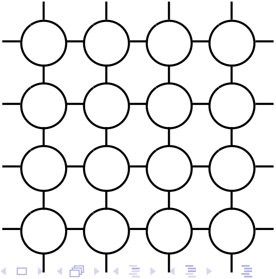
- With sufficient Markovian CI constraints (directed or undirected):

$$P(x_1, \ldots, x_n) \propto \prod_j \Phi_j(\boldsymbol{x}_{N_j}), \quad |N_j| \ll n$$

Can store that. But what about computation?

- Short answer: It depends on global graph structure properties, beyond local factorization

Storage: Linear in n
Computation: Exponential in $n^{1/2}$ [P$\neq$NP]



Node Elimination

Chain:

$$P(x_1, \dots, x_7) = \Phi_1(x_1, x_2)\Phi_2(x_2, x_3) \dots \Phi_6(x_6, x_7)$$

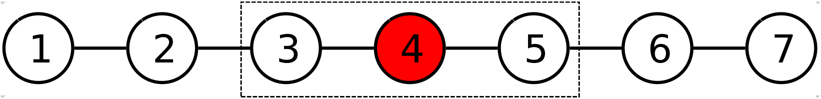
Node Elimination

Chain:

$$P(x_1, \ldots, x_7) = \Phi_1(x_1, x_2)\Phi_2(x_2, x_3) \ldots \Phi_6(x_6, x_7)$$

$$\sum_{x_4} P(x_1, \ldots, {\color{red}x_4}, \ldots, x_7)$$

Node Elimination



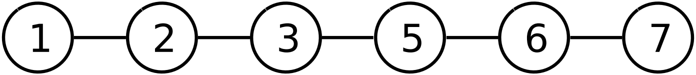
Chain:

$$P(x_1, \ldots, x_7) = \Phi_1(x_1, x_2)\Phi_2(x_2, x_3)\ldots\Phi_6(x_6, x_7)$$

$$\sum_{x_4} P(x_1, \ldots, x_4, \ldots, x_7)$$

$$= \Phi_1(x_1, x_2)\Phi_2(x_2, x_3)\left(\sum_{x_4} \Phi_3(x_3, x_4)\Phi_4(x_4, x_5)\right)\Phi_5(x_5, x_6)\Phi_6(x_6, x_7)$$

Node Elimination



Chain:

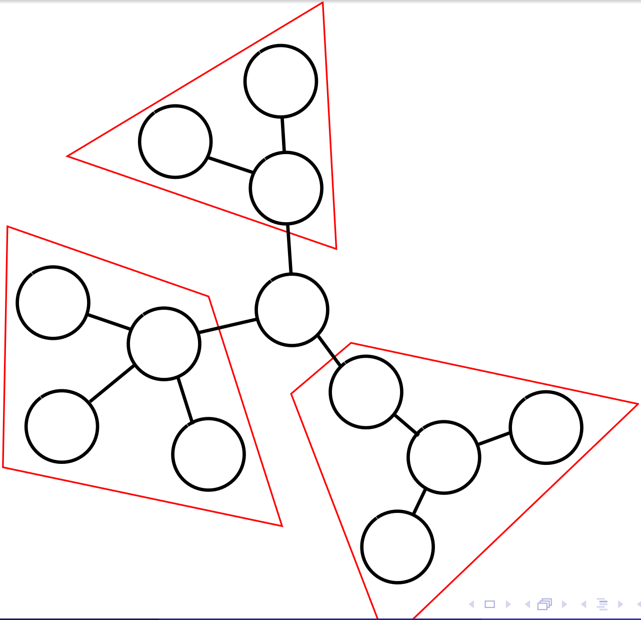
$$P(x_1, \dots, x_7) = \Phi_1(x_1, x_2)\Phi_2(x_2, x_3)\dots\Phi_6(x_6, x_7)$$

$$\sum_{x_4} P(x_1, \dots, {\color{red}x_4}, \dots, x_7)$$

$$= \Phi_1(x_1, x_2)\Phi_2(x_2, x_3)\left(\sum_{x_4}\Phi_3(x_3, {\color{red}x_4})\Phi_4({\color{red}x_4}, x_5)\right)\Phi_5(x_5, x_6)\Phi_6(x_6, x_7)$$

$$= \Phi_1(x_1, x_2)\Phi_2(x_2, x_3)M_{35}(x_3, x_5)\Phi_5(x_5, x_6)\Phi_6(x_6, x_7)$$

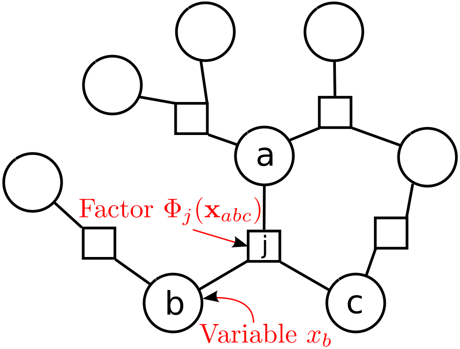
Tree Graphs



Factor Graphs

Factor graphs: Yet another type of graphical model

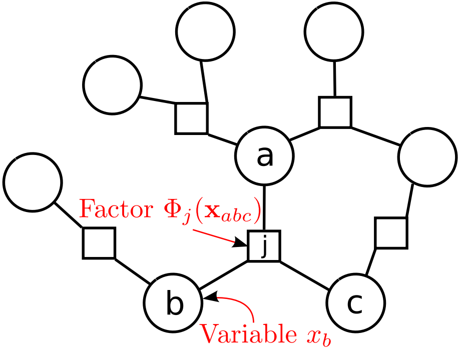
- Bipartite graph: variable / factor nodes
- No probability semantics
- Just for deriving Markovian propagation algorithms
- Factor graph $=$ tree $\Rightarrow$ Fast computation



Factor Graphs

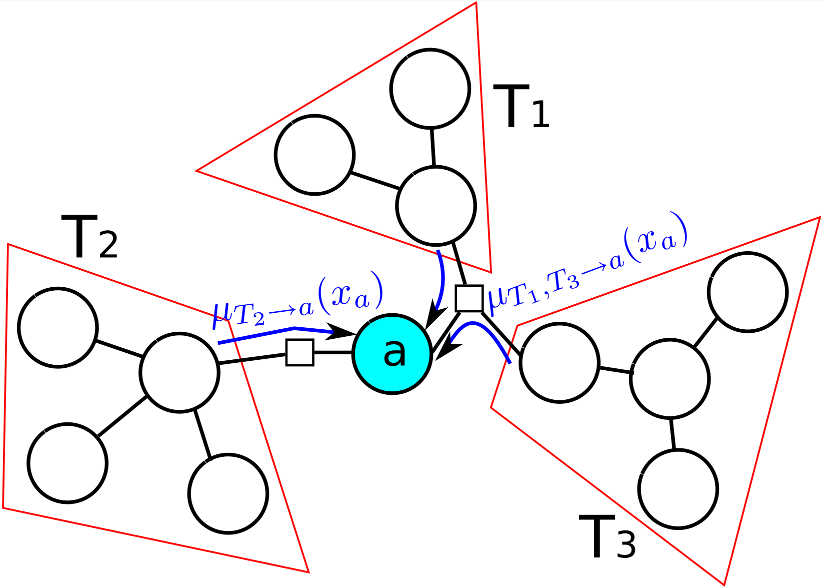
Factor graphs: Yet another type of graphical model

- Bipartite graph: variable / factor nodes
- No probability semantics
- Just for deriving Markovian propagation algorithms
- Factor graph $=$ tree $\Rightarrow$ Fast computation

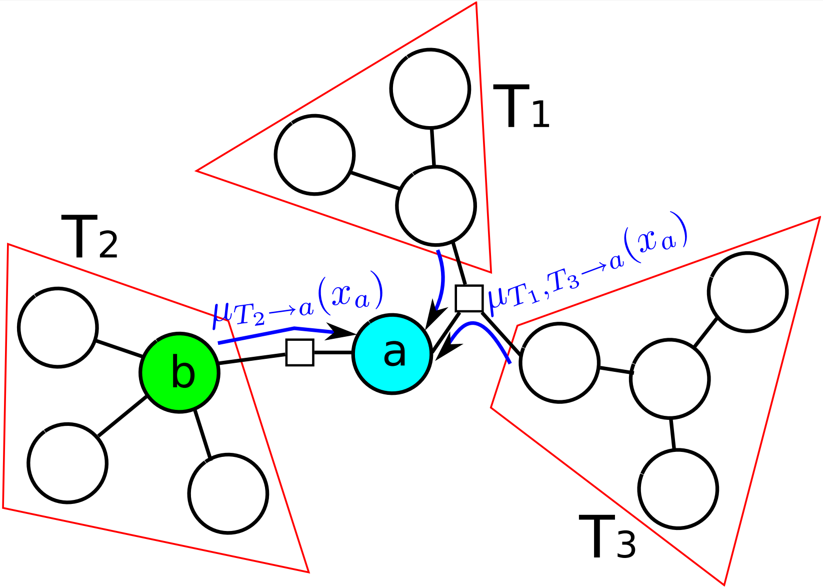


Undirected GM $\rightarrow$ Factor graph: Immediate
Directed GM $\rightarrow$ Factor graph: Easy exercise

Towards Belief Propagation



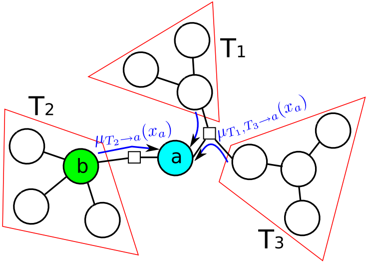
What is a Message?



What is a Message?

- Formally: Directed potential over one variable
- Intuition: Message $T_2 \to a$: What T_2 thinks x_a should be
- Naive "definition":
  - Product: All T_2, and edge $\to a$
  - Sum: All except x_a

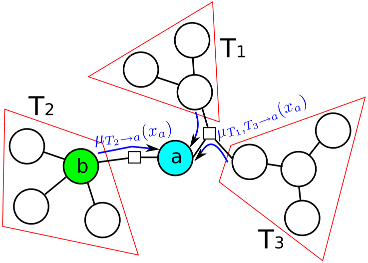
$\Rightarrow$ Real definition recursive ($\mathcal{G}$ tree!)



What is a Message?

- Formally: Directed potential over one variable
- Intuition: Message $T_2 \to a$: What T_2 thinks x_a should be
- Naive "definition":
  - Product: All T_2, and edge $\to a$
  - Sum: All except x_a

$\Rightarrow$ Real definition recursive ($\mathcal{G}$ tree!)



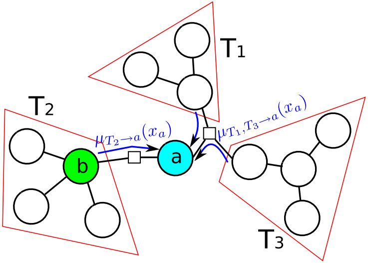
Subtle points:

- Messages: Not conditional / marginal distributions of P. Message $\mu_{T_2 \to a}(x_a)$ has seen T_2 only

What is a Message?

- Formally: Directed potential over one variable
- Intuition: Message $T_2 \to a$: What T_2 thinks x_a should be
- Naive "definition":
  - Product: All T_2, and edge $\to a$
  - Sum: All except x_a

  $\Rightarrow$ Real definition recursive ($\mathcal{G}$ tree!)

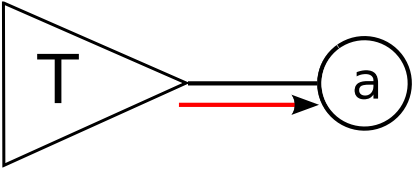


Subtle points:

- Messages: Not conditional / marginal distributions of P. Message $\mu_{T_2 \to a}(x_a)$ has seen T_2 only
- Strictly speaking: Two types of messages: $\bigcirc \to \square$, $\square \to \bigcirc$
  $\Rightarrow$ Understand idea, behind formalities

Message Passing: The Recipe

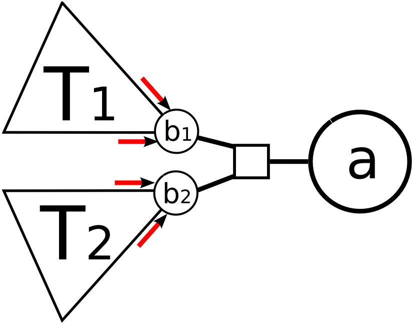
Message $\mu_{T \to a}(x_a)$ to a



Message Passing: The Recipe

Message $\mu_{T \to a}(x_a)$ to a

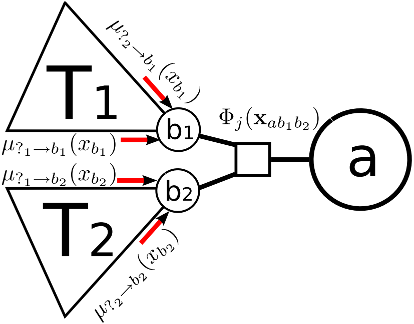
1. Expand factor: (T_1, b_1), (T_2, b_2)



Message Passing: The Recipe

Message $\mu_{T \to a}(x_a)$ to a

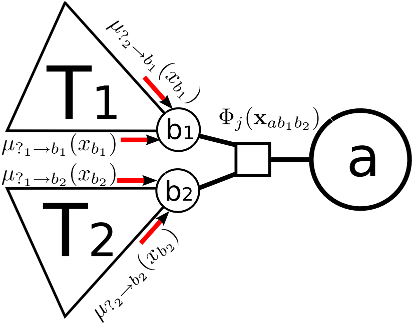
1. Expand factor: (T_1, b_1), (T_2, b_2)
2. Product: Gather potentials
   - $\Phi_j(x_a, x_{b_1}, x_{b_2})$
   - All $\mu_{? \to b_1}(x_{b_1})$, except $b_1 \leftarrow \Phi_j$
   - $\mu_{? \to b_2}(x_{b_2})$ dito



Message Passing: The Recipe

Message $\mu_{T\to a}(x_a)$ to a

1. Expand factor: (T_1, b_1), (T_2, b_2)
2. Product: Gather potentials
   - $\Phi_j(x_a, x_{b_1}, x_{b_2})$
   - All $\mu_{?\to b_1}(x_{b_1})$, except $b_1 \leftarrow \Phi_j$
   - $\mu_{?\to b_2}(x_{b_2})$ dito
3. Sum: Over x_{b_1}, x_{b_2}



$$\mu_{T\to a}(x_a) \propto \sum_{x_{b_1}, x_{b_2}} \Phi_j(\boldsymbol{x}_{ab_1b_2}) \left(\prod_{\tilde{T}:T_1\setminus b_1} \mu_{\tilde{T}\to b_1}(x_{b_1}) \right) \left(\prod_{\tilde{T}:T_2\setminus b_2} \mu_{\tilde{T}\to b_2}(x_{b_2}) \right)$$

Message Passing: The Idea

I can never remember these message passing equations.
What I remember:

- Messages are partial information, given part of graph

Message Passing: The Idea

I can never remember these message passing equations.
What I remember:

- Messages are partial information, given part of graph
- Message passing is information propagation
  - Product: Predict
  - Sum: Marginalize (cover your tracks)

Directed like filtering
$\Rightarrow$ We'll see cases where $\prod$ is not $\prod$, and $\sum$ is not $\sum$

Message Passing: The Idea

I can never remember these message passing equations.
What I remember:

- Messages are partial information, given part of graph
- Message passing is information propagation
    - Product: Predict
    - Sum: Marginalize (cover your tracks)

  Directed like filtering

  $\Rightarrow$ We'll see cases where $\prod$ is not $\prod$, and $\sum$ is not $\sum$
- Marginal distributions (our goal!) are obtained by combining messages $\leftrightarrow$ combining information from all parts

Message Passing: The Idea

I can never remember these message passing equations.
What I remember:

- Messages are partial information, given part of graph
- Message passing is information propagation
    - Product: Predict
    - Sum: Marginalize (cover your tracks)

  Directed like filtering

  $\Rightarrow$ We'll see cases where $\prod$ is not $\prod$, and $\sum$ is not $\sum$
- Marginal distributions (our goal!) are obtained by combining messages $\leftrightarrow$ combining information from all parts
- MP works on trees, because information cannot go around in cycles

Belief Propagation: More than Node Elimination

- Marginalization by message passing:

$$P(x_a) = \sum_{\boldsymbol{x} \setminus x_a} P(\boldsymbol{x}) \propto \Phi_a(x_a) \prod_{j \in \mathcal{N}_a} \mu_{T_j \to a}(x_a)$$

$\mathcal{N}_a$: Factor nodes neighbouring $a \leftrightarrow$ factors $\Phi_j(x_a, \dots)$
Φ_a: Can be $\equiv 1$

Belief Propagation: More than Node Elimination

- Marginalization by message passing:

$$P(x_a) = \sum_{\boldsymbol{x} \setminus x_a} P(\boldsymbol{x}) \propto \Phi_a(x_a) \prod_{j \in \mathcal{N}_a} \mu_{T_j \to a}(x_a)$$

  $\mathcal{N}_a$: Factor nodes neighbouring $a \leftrightarrow$ factors $\Phi_j(x_a, \dots)$
  Φ_a: Can be $\equiv 1$
- All marginals $P(x_1)$, $P(x_2)$, …? Do this n times. Right?
  $\Rightarrow$ NO! Do this twice only!
  $\Rightarrow$ If you understand that, you've understood belief propagation

Belief Propagation: More than Node Elimination

- Marginalization by message passing:

$$P(x_a) = \sum_{\boldsymbol{x} \setminus x_a} P(\boldsymbol{x}) \propto \Phi_a(x_a) \prod_{j \in \mathcal{N}_a} \mu_{T_j \to a}(x_a)$$

  $\mathcal{N}_a$: Factor nodes neighbouring $a \leftrightarrow$ factors $\Phi_j(x_a, \dots)$
  Φ_a: Can be $\equiv 1$
- All marginals $P(x_1)$, $P(x_2)$, $\dots$? Do this n times. Right?
  $\Rightarrow$ NO! Do this twice only!
  $\Rightarrow$ If you understand that, you've understood belief propagation

Belief Propagation on Trees

- Message uniquely defined, independent of use, order of computation
- Message can be computed once all inputs received. Once computed, it does not change anymore
- Compute all messages (2 per edge) $\Rightarrow$ All marginals, O(1) each

Implementation of Belief Propagation

Belief Propagation (Sum-Product) on Trees

1. Designate node (any will do!) as root
2. Inward pass: Compute messages leaves $\rightarrow$ root
3. Outward pass: Compute messages root $\rightarrow$ leaves

Implementation of Belief Propagation

Belief Propagation (Sum-Product) on Trees

1. Designate node (any will do!) as root
2. Inward pass: Compute messages leaves $\to$ root
3. Outward pass: Compute messages root $\to$ leaves

Messages can be normalized at will:

$$\mu_{T\to a}(x_a) = \sum\nolimits_{\boldsymbol{x}_{C_j \setminus a}} \Phi_j(\boldsymbol{x}_{C_j}) \prod C \mu_{\tilde{T}\to b_1}(x_{b_1}) \dots$$

Implementation of Belief Propagation

Belief Propagation (Sum-Product) on Trees

1. Designate node (any will do!) as root
2. Inward pass: Compute messages leaves $\to$ root
3. Outward pass: Compute messages root $\to$ leaves

Messages can be normalized at will:

$$\mu_{T\to a}(x_a) = C \sum_{\boldsymbol{x}_{C_j \setminus a}} \Phi_j(\boldsymbol{x}_{C_j}) \prod \mu_{\tilde{T}\to b_1}(x_{b_1}) \ldots$$

Implementation of Belief Propagation

Belief Propagation (Sum-Product) on Trees

1. Designate node (any will do!) as root
2. Inward pass: Compute messages leaves $\rightarrow$ root
3. Outward pass: Compute messages root $\rightarrow$ leaves

Avoiding underflow / overflow (yes, it does matter):

- Renormalize each message to sum to 1
- Better: Work in log domain (log-messages, log-potentials):

$\prod \rightarrow +$
$\sum \rightarrow$ logsumexp [careful with zeros!]

$$\mathrm{logsumexp}(\boldsymbol{v}) := \log \sum_{i=1}^{k} e^{v_i} = \underbrace{M + \log \sum_{i=1}^{k} e^{v_i - M}}_{\text{numerically stable}}, \quad M = \max_i v_i$$

Searching for the Mode: Max-Product

Decoding:

$$\boldsymbol{x}_* \in \underset{\boldsymbol{x}}{\operatorname{argmax}}\, P(\boldsymbol{x})$$

max, $\prod$: Same decomposition as $\sum, \prod$.
Better: max, $\sum$ in log domain

Searching for the Mode: Max-Product

Decoding:

$$\boldsymbol{x}_* \in \operatorname*{argmax}_{\boldsymbol{x}} P(\boldsymbol{x})$$

max, $\prod$: Same decomposition as $\sum, \prod$.
Better: max, $\sum$ in log domain

- Max-messages:

$$\mu_{T\to a}(x_a) = \max_{\boldsymbol{x}_{C_j \setminus a}} \left(\log \Phi_j(\boldsymbol{x}_{C_j}) + \sum\nolimits_{b \in C_j \setminus a} \mu_{T_b \to b}(x_b) \right)$$

Searching for the Mode: Max-Product

Decoding:

$$\boldsymbol{x}_* \in \operatorname*{argmax}_{\boldsymbol{x}} P(\boldsymbol{x})$$

max, $\prod$: Same decomposition as $\sum, \prod$.
Better: max, $\sum$ in log domain

- Max-messages:

$$\mu_{T\to a}(x_a) = \max_{\boldsymbol{x}_{C_j\setminus a}} \left(\log \Phi_j(\boldsymbol{x}_{C_j}) + \sum\nolimits_{b\in C_j\setminus a} \mu_{T_b\to b}(x_b) \right)$$

- Back-pointer tables:

$$\delta_{T\to a}(x_a) \in \operatorname*{argmax}_{\boldsymbol{x}_{C_j\setminus a}} \left(\log \Phi_j(\boldsymbol{x}_{C_j}) + \sum\nolimits_{b\in C_j\setminus a} \mu_{T_b\to b}(x_b) \right)$$

Wrap-Up

- Belief propagation (sum-product) on trees:
  All marginals in linear time, by local information propagation
- Max-product, max-sum, logsumexp-sum, ...:
  What matters is the graph!

Wrap-Up

- Belief propagation (sum-product) on trees:
  All marginals in linear time, by local information propagation
- Max-product, max-sum, logsumexp-sum, . . . :
  What matters is the graph!
- What about general graphs?
  - Decomposable graphs. Treewidth of a graph
  - Junction tree algorithm

  Interested?
  - PMR Edinburgh slides:

    `http://www.inf.ed.ac.uk/teaching/courses/pmr/slides/jta-2x2.pdf`
  - Lauritzen, S; Spiegelhalter, D. Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems. JRSS-B, 50: 157-224 (1988)

Wrap-Up

- Belief propagation (sum-product) on trees:
  All marginals in linear time, by local information propagation
- Max-product, max-sum, logsumexp-sum, . . . :
  What matters is the graph!
- What about general graphs?
  - Decomposable graphs. Treewidth of a graph
  - Junction tree algorithm

  Interested?
  - PMR Edinburgh slides:
    `http://www.inf.ed.ac.uk/teaching/courses/pmr/slides/jta-2x2.pdf`
  - Lauritzen, S; Spiegelhalter, D. Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems. JRSS-B, 50: 157-224 (1988)
- Beware (not surprising): Inference on general graphs is NP hard.
  In general, approximations are a must