Universität Potsdam
Institut für Informatik
Lehrstuhl Maschinelles Lernen

Graphical Models

Niels Landwehr

Agenda

- Graphical models: syntax and semantics.
- Inference in graphical models (exact, approximate)
- Graphical models in machine learning.

Agenda

- Graphical models: syntax and semantics.
- Inference in graphical models (exact, approximate)
- Graphical models in machine learning.

Problem Setting Inference

- Given: graphical model over random variables $\{X_1, \ldots, X_N\}$.

- Problem setting inference:
  - Variables with evidence $X_{i_1}, \ldots, X_{i_m}$ $\quad \{i_1, \ldots, i_m\} \subseteq \{1, \ldots, N\}$
  - Query variable X_a $\quad a \in \{1, \ldots, N\} \setminus \{i_1, \ldots, i_m\}$

  - Task: compute distribution over query variable given evidence.

Conditional distribution over random variable X_a

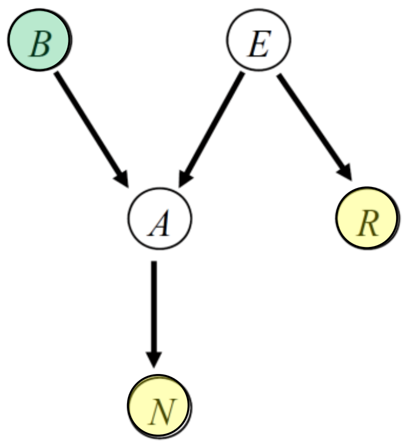
Evidence: observed values for variables $X_{i_1}, \ldots, X_{i_m}$

Compute $\quad p(x_a \mid x_{i_1}, \ldots, x_{i_m})$

More generally also $\quad p(x_{a_1}, \ldots, x_{a_k} \mid x_{i_1}, \ldots, x_{i_m})$

Graphical models: inference

- Example „Alarm“ domain
  - Variables with evidence: N, R
  - Query variable: B



Probability for burglary given

that neighbor has called and radio report yes/no?

For example:

$$p(B=1 \mid N=1, R=0) = 0.7$$
$$p(B=0 \mid N=1, R=0) = 0.3$$

$$p(B=1 \mid N=1, R=1) = 0.2$$
$$p(B=0 \mid N=1, R=1) = 0.8$$

- Posterior over parameters, Bayesian prediction, …

Graphical Models: Inference

- Inference a difficult problem in general
  - General graphical models: exact inference is NP-hard.
  - There are algorithms for exact inference in general graphical models whose execution time depends on properties of the graph structure („Message-Passing“)
  - There are several techniques for approximate inference (Sampling, Variational Inference, Expectation Propagation).

- We will look at
  - Message-Passing algorithm: special cases.
  - Sampling-based approximate inference.

Inference: Discrete vs. Continuous Variables

- We will discuss inference only for discrete variables.

- Discussed inference algorithms are also applicable to continuous variables
  - Replace sums by integrals
  - Distribution families have to be chosen in such a way that integrals can indeed be computed (in closed form).

Agenda

- Graphical models: syntax and semantics.
- Inference in graphical models
  - Exact inference
  - Approximate inference
- Graphical models in machine learning.

Exact Inference: Naive Computation

- Graphical model: representation of $p(X_1,\ldots,X_N)$.

- Naive inference computation:

$$\text{Notation}: \{X_1,\ldots,X_N\} = \{ \underbrace{X_a}_{\text{query variable}}, \underbrace{X_{i_1},\ldots,X_{i_m}}_{\text{evidence variables}}, \underbrace{X_{j_1},\ldots,X_{j_k}}_{\text{remaining variables}} \}$$

Compute for each value x_a: $$p(x_a \mid x_{i_1},\ldots,x_{i_m}) = \frac{p(x_a, x_{i_1},\ldots,x_{i_m})}{p(x_{i_1},\ldots,x_{i_m})}$$

$$= \frac{1}{Z} p(x_a, x_{i_1},\ldots,x_{i_m})$$

$$= \frac{1}{Z} \sum_{x_{j1}} \sum_{x_{j2}} \cdots \sum_{x_{jk}} p(x_1,\ldots,x_N)$$

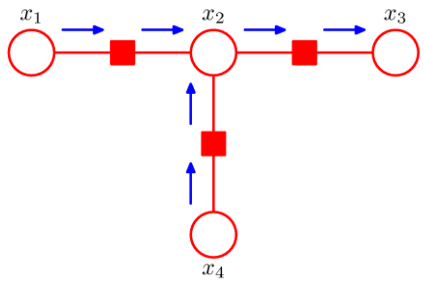
Z is a normalizer, easy to compute explicitly for univariate distributions

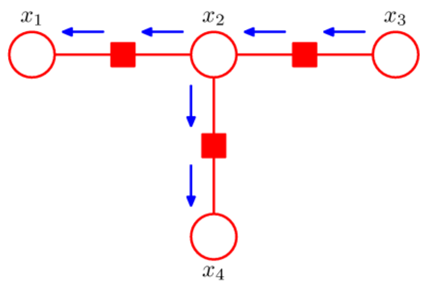
Central problem: summing out all remaining variables (exponential time if done naively)

Intelligent Data Analysis II

More Efficient Inference

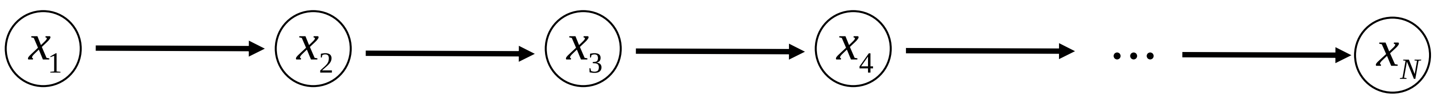
- More efficient method than naive inference computation?
  - For general graphs probably impossible (NP-hard problem).
  - But if there is the right structure in the model (independencies), we can potentially exploit this structure to speed up inference.
- Idea: Local computations that are propagated along the graph structure
  - Nodes send each other „messages“ that contain results of partial calculations.
  - „Message Passing“, „Belief Propagation“.
  - Execution time of the methods depends on the graph structure (exponential in worst case).





Graphical Model: Inference on Linear Chain

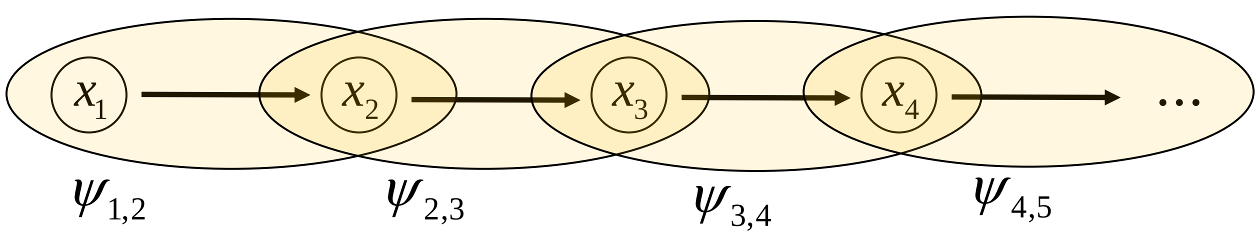
- We now study the Message Passing algorithm in a special case with particularly simple structure: linear chain of random variables.



$$p(x_1,\dots,x_N) = p(x_1)\,p(x_2 \mid x_1)\,p(x_3 \mid x_2)\cdot\ldots\cdot p(x_N \mid x_{N-1})$$

- Notation: represent the joint distribution as a product of potential functions over pairs of random variables.

$$p(x_1,\dots,x_N) = \overbrace{\psi_{1,2}(x_1,x_2)}^{p(x_1)\,p(x_2|x_1)}\,\overbrace{\psi_{2,3}(x_2,x_3)}^{p(x_3|x_2)}\cdot\ldots\cdot\overbrace{\psi_{N-1,N}(x_{N-1},x_N)}^{p(x_N|x_{N-1})}$$



Inference: Linear Chain of Random Variables

- Introduction of Message Passing algorithm by an example.
- Linear chain of 5 random variables:

$$x_1 \longrightarrow x_2 \longrightarrow x_3 \longrightarrow x_4 \longrightarrow x_5$$

- Compute marginal distribution of 3. variable (without evidence).

query variable

remaining variables (being summed out)

$$p(x_3) = \sum_{x_1} \sum_{x_2} \sum_{x_4} \sum_{x_5} p(x_1, x_2, x_3, x_4, x_5)$$

$$= \sum_{x_1} \sum_{x_2} \sum_{x_4} \sum_{x_5} \psi_{1,2}(x_1, x_2) \psi_{2,3}(x_2, x_3) \psi_{3,4}(x_3, x_4) \psi_{4,5}(x_4, x_5)$$

- Naive computation exponential (because of nested sums).
- Idea: exploit structure (linear chain) for more efficient computation.

Inference: Message Passing

- Exploit factorization of joint distribution into potentials (that is, exploit independence assumptions encoded in chain).

$$p(x_3) = \sum_{x_1} \sum_{x_2} \sum_{x_4} \sum_{x_5} \psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\psi_{3,4}(x_3, x_4)\psi_{4,5}(x_4, x_5)$$
$$= \sum_{x_1} \sum_{x_2} \sum_{x_4} \psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\psi_{3,4}(x_3, x_4) \underbrace{\sum_{x_5} \psi_{4,5}(x_4, x_5)}_{\text{local, partial computation: "message" } \mu_\beta(x_4)}$$

- Local partial computation: „message“ $\mu_\beta(x_4)$
  - Compute for all values of x_4: $\mu_\beta(x_4) = \sum_{x_5} \psi_{4,5}(x_4, x_5)$
  - Message is function of the state of variable x_4 (coded e.g. as a vector whose elements are message values for different states).
  - In the message the node X_5 has been summed out.

Inference: Message Passing

- Exploit factorization of joint distribution into potentials (that is, exploit independence assumptions encoded in chain).

$$p(x_3) = \sum_{x_1} \sum_{x_2} \sum_{x_4} \sum_{x_5} \psi_{1,2}(x_1, x_2) \psi_{2,3}(x_2, x_3) \psi_{3,4}(x_3, x_4) \psi_{4,5}(x_4, x_5)$$

$$= \sum_{x_1} \sum_{x_2} \sum_{x_4} \psi_{1,2}(x_1, x_2) \psi_{2,3}(x_2, x_3) \psi_{3,4}(x_3, x_4) \underbrace{\sum_{x_5} \psi_{4,5}(x_4, x_5)}_{}$$

local, partial computation: "message" $\mu_\beta(x_4)$

- Local partial

  - Compute fo

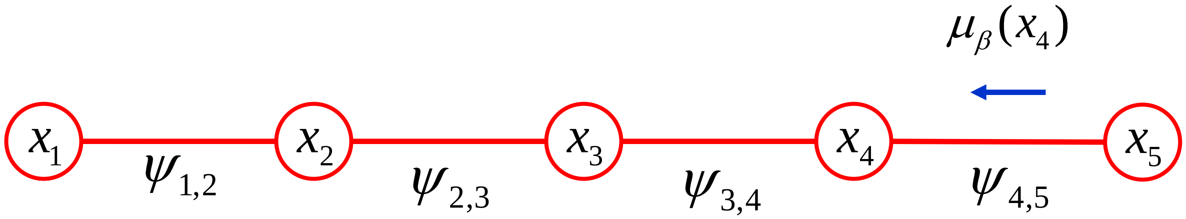
x_5

Coding as vector: $\mu_\beta(x_4) = \begin{pmatrix} \sum_{x_5} \psi_{4,5}(0, x_5) \\ \sum_{x_5} \psi_{4,5}(1, x_5) \end{pmatrix}$

  - Message is fun of the state of variable x_4 (coded e.g. as a vector whose elements are message values for different states).
  - In the message the node X_5 has been summed out.

Inference: Message Passing

- **Intuition**: We sum out the node X_5 , and send the result along to node X_4 .



Inference: Message Passing

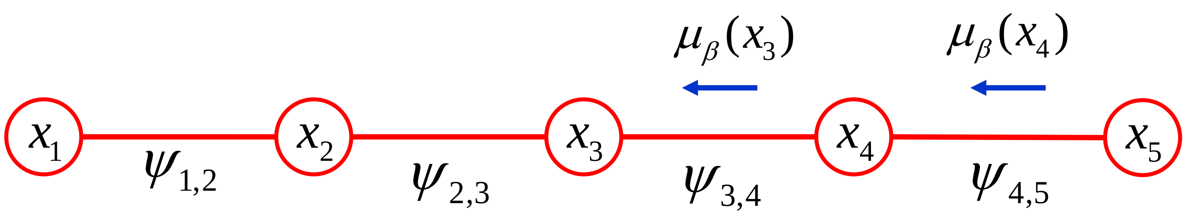
- We apply the same idea to the next variable that has to be summed out:

$$p(x_3) = \sum_{x_1} \sum_{x_2} \sum_{x_4} \psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\psi_{3,4}(x_3, x_4)\mu_\beta(x_4)$$
$$= \sum_{x_1} \sum_{x_2} \psi_{1,2}(x_1, x_2)\psi_{2,3}(x_2, x_3)\underbrace{\sum_{x_4} \psi_{3,4}(x_3, x_4)\mu_\beta(x_4)}_{\text{Local partial computation: "message" } \mu_\beta(x_3)}$$

- Local partial computation: „message“ $\mu_\beta(x_3)$
  - Compute for all values of x_3: $\mu_\beta(x_3) = \sum_{x_4} \psi_{3,4}(x_3, x_4)\mu_\beta(x_4)$
  - Message is function of the state of variable x_3
  - In the message, the nodes X_5, X_4 have been summed out.

Inference: Message Passing

- **Intuition**: We sum out the node X_4 , and send the result along to node X_3 .



- X_3 is query node, so we do not want to sum it out…

Inference: Message Passing

- Apply the same idea to the variables to the left of the query variable

$$p(x_3) = \sum_{x_1} \sum_{x_2} \psi_{1,2}(x_1, x_2) \psi_{2,3}(x_2, x_3) \mu_\beta(x_3)$$

change summation order, rearrange terms

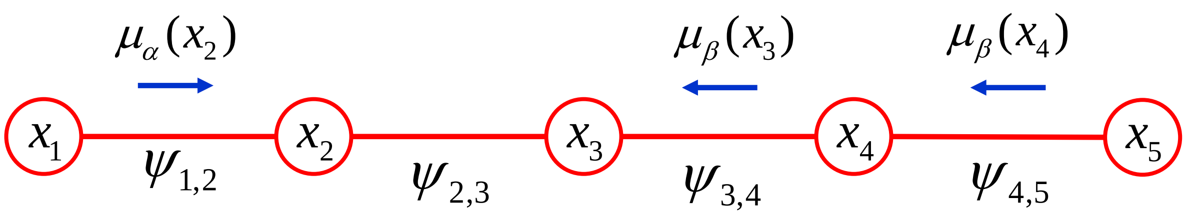
$$= \mu_\beta(x_3) \sum_{x_2} \sum_{x_1} \psi_{2,3}(x_2, x_3) \psi_{1,2}(x_1, x_2)$$

$$= \mu_\beta(x_3) \sum_{x_2} \psi_{2,3}(x_2, x_3) \underbrace{\sum_{x_1} \psi_{1,2}(x_1, x_2)}_{\text{"message" } \mu_\alpha(x_2)}$$

- Local partial computation: „message" $\mu_\alpha(x_2)$
  - Compute for all values of x_2: $\mu_\alpha(x_2) = \sum_{x_1} \psi_{1,2}(x_1, x_2)$
  - Message is function of the state of variable x_2.
  - In the message the node X_1 has been summed out.

Inference: Message Passing

- **Intuition**: We sum out the node X_1 , and send the result along to node X_2 .



Inference: Message Passing

- Summing out last variable X_2:

$$p(x_3) = \mu_\beta(x_3) \underbrace{\sum_{x_2} \psi_{2,3}(x_2, x_3)\mu_\alpha(x_2)}_{\text{"message" } \mu_\alpha(x_3)}$$

$$= \mu_\beta(x_3)\mu_\alpha(x_3)$$

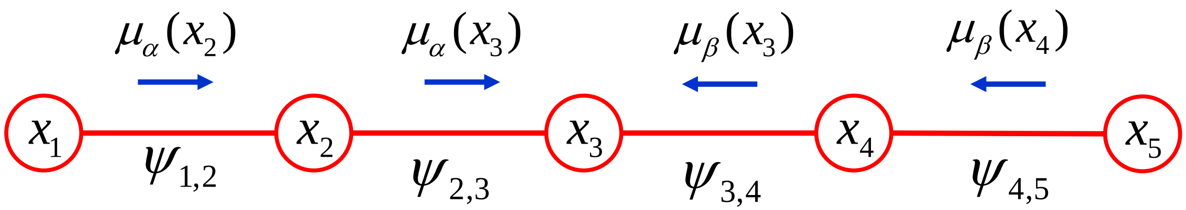
  - Result: marginal distribution we wanted to compute is product of messages arriving at query node:

$$p(x_3) = \mu_\beta(x_3)\mu_\alpha(x_3)$$

  - The messages are a function of x_3, so this gives us a distribution.

Inference: Message Passing

- Schema of how to pass messages:



$$p(x_3) = \mu_\beta(x_3)\mu_\alpha(x_3)$$

Final result: marginal distribution is product of messages.

Inference: Message Passing

- Execution time:
  - Computation of a single message:

    Compute for all values of x_3: $\mu_\beta(x_3) = \sum_{x_4} \psi_{3,4}(x_3, x_4)\mu_\beta(x_4)$

    $\Rightarrow O(M^2)$ for computation of a message (assuming variables with M discrete states)

  - $N\text{-}1$ messages overall

    $\Rightarrow O(NM^2)$ total running time.

  - Much better than naive inference which takes time $O(M^N)$.

Intelligent Data Analysis II

Inference: Message Passing Algorithm

- Algorithm: Message Passing on a linear chain
  - Input:

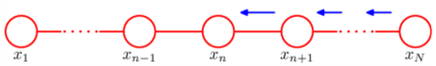
$$p(x_1,\ldots,x_N) = \psi_{1,2}(x_1,x_2),\ldots,\psi_{N-1,N}(x_{N-1},x_N)$$

Query: $p(x_a) = ?$

  - Recursively compute messages:

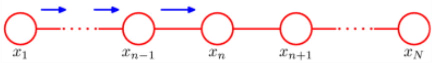
$$\mu_\beta(x_N) = \mathbf{1}$$

For $k = N\text{-}1,\ldots,a$: $\quad \mu_\beta(x_k) = \sum_{x_{k+1}} \psi_{k,k+1}(x_k,x_{k+1})\mu_\beta(x_{k+1})$



$$\mu_\alpha(x_1) = \mathbf{1}$$

For $k = 2,\ldots,a$: $\quad \mu_\alpha(x_k) = \sum_{x_{k-1}} \psi_{k-1,k}(x_{k-1},x_k)\mu_\alpha(x_{k-1})$



  - Output:

$p(x_a) = \mu_\alpha(x_a)\mu_\beta(x_a)$ (function of x_a, that is, distribution over x_a)

Message Passing with Evidence

- So far we have computed marginal $p(x_a)$ without evidence.
- What about conditional distributions given evidence?

$$\text{Notation}: \{X_1, \ldots, X_N\} = \{\underbrace{X_a}_{\text{query variable}}, \underbrace{X_{i_1}, \ldots, X_{i_m}}_{\text{evidence variables}}, \underbrace{X_{j_1}, \ldots, X_{j_k}}_{\text{remaining variables}}\}$$

- Goal: Conditional distribution

$$p(x_a \mid x_{i_1}, \ldots, x_{i_m}) = \frac{p(x_a, x_{i_1}, \ldots., x_{i_m})}{p(x_{i_1}, \ldots., x_{i_m})}$$

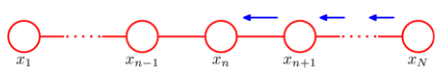
$$= \frac{1}{Z} p(x_a, x_{i_1}, \ldots., x_{i_m})$$

- Z is easy to compute: normalizer of a univariate distribution.
- Therefore we need to compute $p(x_a, x_{i_1}, \ldots., x_{i_m})$.

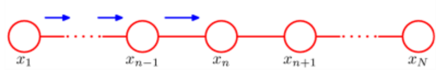
Message Passing with Evidence

- Goal: $p(x_a, x_{i_1}, \ldots, x_{i_m}) = ?$
- Slight modification of Message Passing algorithm
  - Like in the old version, we still compute messages

$$\mu_\beta(x_{N-1}), \ldots, \ \mu_\beta(x_a)$$



$$\mu_\alpha(x_2), \ldots, \ \mu_\alpha(x_a)$$



  - If x_{k+1} is not observed, we sum out this node:

$$k+1 \notin \{i_1, \ldots, i_m\} \quad \Rightarrow \quad \mu_\beta(x_k) = \sum_{x_{k+1}} \psi_{k,k+1}(x_k, x_{k+1}) \mu_\beta(x_{k+1})$$

  - If x_{k+1} is observed, we use only the summand corresponding to the observation:

x_{k+1} observed value (evidence)

$$k+1 \in \{i_1, \ldots, i_m\} \quad \Rightarrow \quad \mu_\beta(x_k) = \psi_{k,k+1}(x_k, x_{k+1}) \mu_\beta(x_{k+1})$$

Message Passing with Evidence

- Analogously for $\mu_\alpha(x_k)$

$$\mu_\alpha(x_k) = \begin{cases} \sum_{x_{k-1}} \psi_{k-1,k}(x_{k-1}, x_k)\mu_\alpha(x_{k-1}) : k-1 \notin \{i_1, \ldots, i_m\} & \text{(node not observed)} \\ \psi_{k-1,k}(x_{k-1}, x_k)\mu_\alpha(x_{k-1}) : k-1 \in \{i_1, \ldots, i_m\} & \text{(node observed)} \end{cases}$$

- Now it holds that

$$p(x_a, x_{i_1}, \ldots, x_{i_m}) = \mu_\alpha(x_a)\mu_\beta(x_a).$$

- Execution time for inference with evidence is still $O(NM^2)$.

Example: Markov Models

- Example for inference on linear chain: Markov models.

- Markov model: simple model for dynamic probabilistic process
  - Process that can take on different states
  - Random variable X_t represents state at time t
  - Discrete time steps $t=1,\ldots,T$

- Example: weather
  - Random variable X_t = weather at day t.
  - Two possible states, rain and sunshine.

Example: Markov Models

- Dynamic model:
  - Process is started in a random state:

    Distribution over initial states $p(x_1)$

  - At each time step, the process randomly changes into a new state, where the transition probability only depends on the current state (simplifying assumption!).

    Distribution for state transitions $p(x_{t+1} \mid x_t)$

- Independence assumption

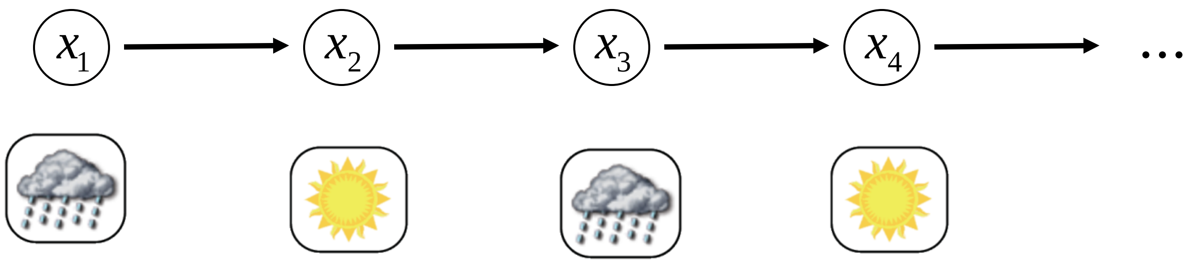
  $\forall t: \; p(x_{t+1} \mid x_1, \ldots, x_t) = p(x_{t+1} \mid x_t)$ "Markov" property

- Transition probabilities do not depend on *t*:

  $\forall t: \; p(x_{t+1} \mid x_t) = p(x_t \mid x_{t-1})$ "Stationary" process

Example: Markov Models

- Example Markov model:
  - State x_t = weather at day t
  - Two possible states, rain and sunshine



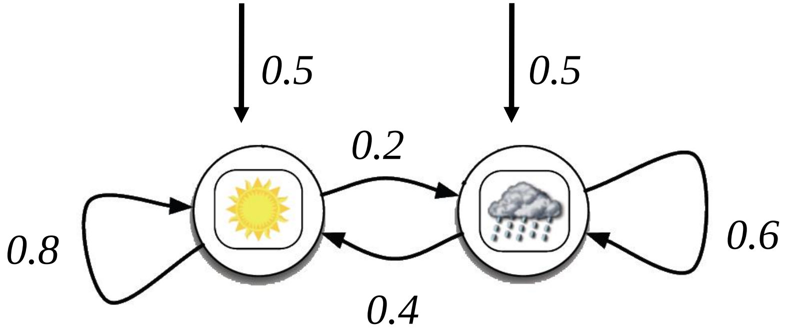
  - Distributions

$$p(x_1 = s) = 0.5$$
$$p(x_1 = r) = 0.5$$

$$p(x_{t+1} = s \mid x_t = s) = 0.8$$
$$p(x_{t+1} = r \mid x_t = s) = 0.2$$
$$p(x_{t+1} = s \mid x_t = r) = 0.4$$
$$p(x_{t+1} = r \mid x_t = r) = 0.6$$

Example: Markov Models

- Markov models correspond to probabilistic finite automata:
  - Start in randomly chosen state
  - At each time step, randomly transition to a novel state, based on the current state.

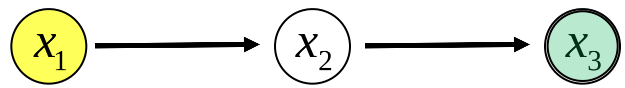


Automaton model

Example: Markov Models

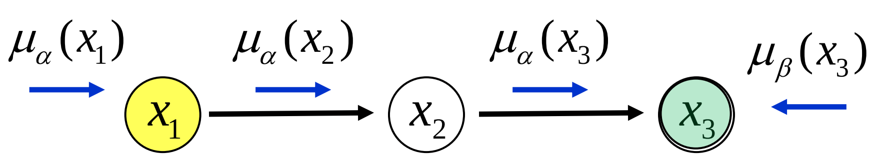
- Example for inference problem:
  - How likely is it that the sun shines the day after tomorrow, given that it rains today?

$$p(x_3 \mid x_1 = r) = ?$$



- Computation with message passing algorithm.

- Messages: $\mu_\alpha(x_1), \mu_\alpha(x_2), \mu_\alpha(x_3); \ \ \mu_\beta(x_3).$



Example: Markov Models

- Computation of messages: according to Slides 28/29, plugging in values from Slide 32.

$$\mu_\alpha(x_1 = s) = 1$$

$$\mu_\alpha(x_1 = r) = 1$$

Initialization

$$\mu_\alpha(x_2 = s) = p(x_1 = r)p(x_2 = s \mid x_1 = r)\mu_\alpha(x_1 = r) = 0.5 \cdot 0.4 \cdot 1 = 0.2$$

$$\mu_\alpha(x_2 = r) = p(x_1 = r)p(x_2 = r \mid x_1 = r)\mu_\alpha(x_1 = r) = 0.5 \cdot 0.6 \cdot 1 = 0.3$$

$$\mu_\alpha(x_3 = s) = p(x_3 = s \mid x_2 = s)\mu_\alpha(x_2 = s) + p(x_3 = s \mid x_2 = r)\mu_\alpha(x_2 = r) = 0.8 \cdot 0.2 + 0.4 \cdot 0.3 = 0.28$$

$$\mu_\alpha(x_3 = r) = p(x_3 = r \mid x_2 = s)\mu_\alpha(x_2 = s) + p(x_3 = r \mid x_2 = r)\mu_\alpha(x_2 = r) = 0.2 \cdot 0.2 + 0.6 \cdot 0.3 = 0.22$$

$$\mu_\beta(x_3 = s) = 1$$

$$\mu_\beta(x_3 = r) = 1$$

Example: Markov Models

- Computation of messages: according to Slides 25/26, plugging in values from Slide 29.

$$\mu_\alpha(x_1 = s) = 1$$

$$\mu_\alpha(x_1 = r) = 1$$

Preceeding node x_1 observed: no summation

$$\mu_\alpha(x_2 = s) = p(x_1 = r)\, p(x_2 = s \mid x_1 = r)\, \mu_\alpha(x_1 = r) = 0.5 \cdot 0.4 \cdot 1 = 0.2$$

$$\mu_\alpha(x_2 = r) = p(x_1 = r)\, p(x_2 = r \mid x_1 = r)\, \mu_\alpha(x_1 = r) = 0.5 \cdot 0.6 \cdot 1 = 0.3$$

$$\mu_\alpha(x_3 = s) = p(x_3 = s \mid x_2 = s)\, \mu_\alpha(x_2 = s) + p(x_3 = s \mid x_2 = r)\, \mu_\alpha(x_2 = r) = 0.8 \cdot 0.2 + 0.4 \cdot 0.3 = 0.28$$

$$\mu_\alpha(x_3 = r) = p(x_3 = r \mid x_2 = s)\, \mu_\alpha(x_2 = s) + p(x_3 = r \mid x_2 = r)\, \mu_\alpha(x_2 = r) = 0.2 \cdot 0.2 + 0.6 \cdot 0.3 = 0.22$$

$$\mu_\beta(x_3 = s) = 1$$

$$\mu_\beta(x_3 = r) = 1$$

Example: Markov Models

- Computation of messages: according to Slides 25/26, plugging in values from Slide 29.

$$\mu_\alpha(x_1 = s) = 1$$
$$\mu_\alpha(x_1 = r) = 1$$

$$\mu_\alpha(x_2 = s) = p(x_1 = r)\,p(x_2 = s \mid x_1 = r)\,\mu_\alpha(x_1 = r) = 0.5 \cdot 0.4 \cdot 1 = 0.2$$
$$\mu_\alpha(x_2 = r) = p(x_1 = r)\,p(x_2 = r \mid x_1 = r)\,\mu_\alpha(x_1 = r) = 0.5 \cdot 0.6 \cdot 1 = 0.3$$

Preceeding node x_2 not observed: sum out

$$\mu_\alpha(x_3 = s) = p(x_3 = s \mid x_2 = s)\,\mu_\alpha(x_2 = s) + p(x_3 = s \mid x_2 = r)\,\mu_\alpha(x_2 = r) = 0.8 \cdot 0.2 + 0.4 \cdot 0.3 = 0.28$$
$$\mu_\alpha(x_3 = r) = p(x_3 = r \mid x_2 = s)\,\mu_\alpha(x_2 = s) + p(x_3 = r \mid x_2 = r)\,\mu_\alpha(x_2 = r) = 0.2 \cdot 0.2 + 0.6 \cdot 0.3 = 0.22$$

$$\mu_\beta(x_3 = s) = 1$$
$$\mu_\beta(x_3 = r) = 1$$

Example: Markov Models

- Computation of messages: according to Slides 25/26, plugging in values from Slide 29.

$$\mu_\alpha(x_1 = s) = 1$$
$$\mu_\alpha(x_1 = r) = 1$$

$$\mu_\alpha(x_2 = s) = p(x_1 = r)p(x_2 = s \mid x_1 = r)\mu_\alpha(x_1 = r) = 0.5 \cdot 0.4 \cdot 1 = 0.2$$
$$\mu_\alpha(x_2 = r) = p(x_1 = r)p(x_2 = r \mid x_1 = r)\mu_\alpha(x_1 = r) = 0.5 \cdot 0.6 \cdot 1 = 0.3$$

$$\mu_\alpha(x_3 = s) = p(x_3 = s \mid x_2 = s)\mu_\alpha(x_2 = s) + p(x_3 = s \mid x_2 = r)\mu_\alpha(x_2 = r) = 0.8 \cdot 0.2 + 0.4 \cdot 0.3 = 0.28$$
$$\mu_\alpha(x_3 = r) = p(x_3 = r \mid x_2 = s)\mu_\alpha(x_2 = s) + p(x_3 = r \mid x_2 = r)\mu_\alpha(x_2 = r) = 0.2 \cdot 0.2 + 0.6 \cdot 0.3 = 0.22$$

$$\mu_\beta(x_3 = s) = 1$$
$$\mu_\beta(x_3 = r) = 1$$

Initialization

Example: Markov Models

- Result: Product of messages arriving at query node.

$$p(x_3 = s \mid x_1 = r) = \frac{1}{Z}\mu_\alpha(x_3 = s)\mu_\beta(x_3 = s) = \frac{1}{Z}0.28$$

$$p(x_3 = r \mid x_1 = r) = \frac{1}{Z}\mu_\alpha(x_3 = r)\mu_\beta(x_3 = r) = \frac{1}{Z}0.22$$

$$Z = 0.28 + 0.22 = 0.5$$

$$p(x_3 = s \mid x_1 = r) = 0.56$$

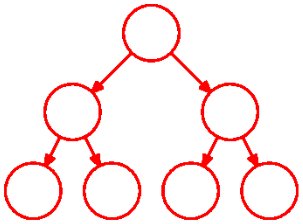
$$p(x_3 = r \mid x_1 = r) = 0.44$$

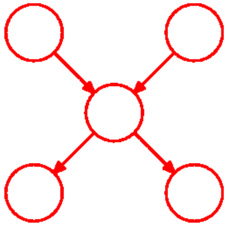
- Likelihood that the sun shines the day after tomorrow, given that it rains today: 56%.

Inference in General Graphs

- So far only looked at special case: inference on linear chain.
- The general idea of message passing also applies to more general graphs.
- Extension: exact inference on *polytrees*
  - Polytree: directed graph in which there is exactly one undirected path between any two nodes.
  - Slightly more general concept than directed tree.

Directed tree





Polytree

Factor Graphs

- General idea for message passing on polytrees: Transformation into *factor graph.*

- Given a graphical model over random variables $\{X_1, \ldots, X_N\}$ with graph structure G.
- The graphical model defines a joint distribution by

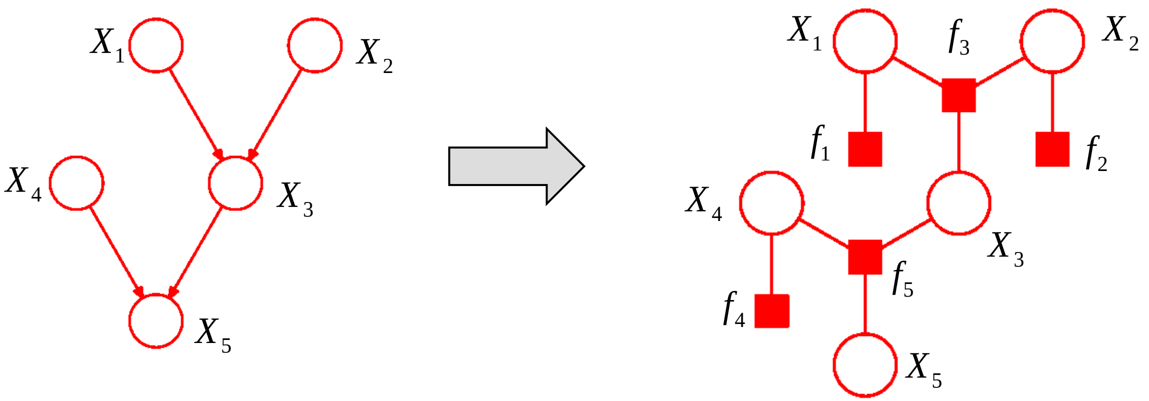
$$p(X_1, \ldots, X_N) = \prod_{i=1}^{N} p(X_i \mid pa(X_i)).$$

- **Definition**: The *factor graph* of the graphical model is a bipartite undirected graph with
  - node set $\{X_1, \ldots, X_N\} \cup \{f_1, \ldots, f_N\}$ (f_i are called *factor nodes*)
  - edge between X_i and f_i for $i = 1, \ldots, N$
  - edge between X_j and f_i if $X_j \in pa(X_i)$.

Factor Graphs: Example

- Factor graph: make factors in joint distribution $\prod_{i=1}^{N} p(X_i \mid pa(X_i))$ explicit.
  - For each variable, there is a variable node (circles).
  - For each factor, there is a factor node (rectangles).
  - Variables are connected to factors they appear in.

Initial graph

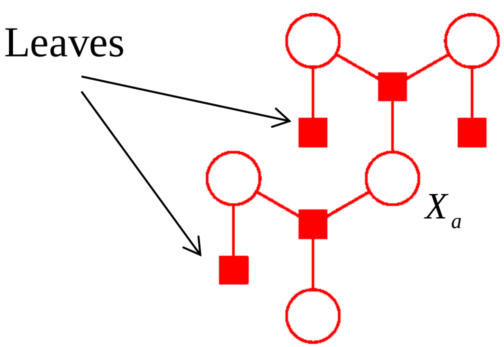


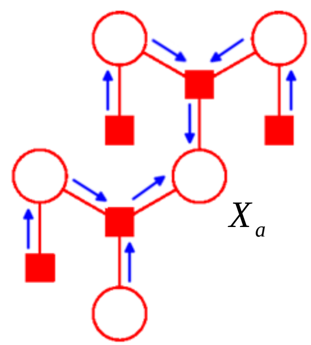
Factor graph

Joint distribution: $p(X_1, X_2, X_3, X_4, X_5) = p(X_1)\,p(X_2)\,p(X_3 \mid X_1, X_2)\,p(X_4)\,\underbrace{p(X_5 \mid X_3, X_4)}_{f_5}$

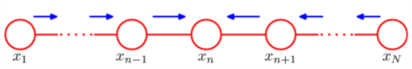
Inference on Factor Graphs

- If the orginial graph was a polytree, the resulting factor graph is an undirected tree (that is, it has no cycles).



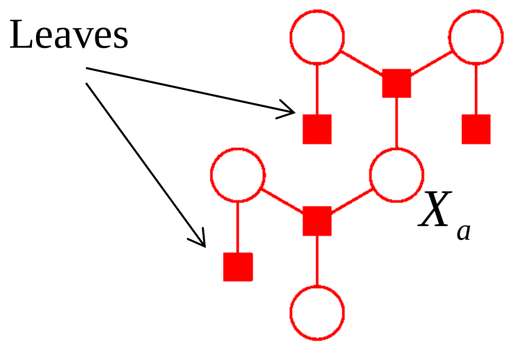


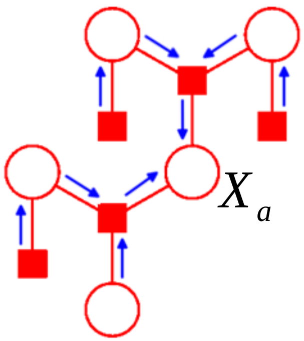
Special case linear chain:



- Inference is then carried out on factor graph:
  - Take the query node X_a as the root of the undirected tree.
  - Send messages from the leaves to the root (there is always a unique path, because factor graph is undirected tree).
  - There are now two types of messages: factor messages and variable messages.

Inference on Factor Graphs

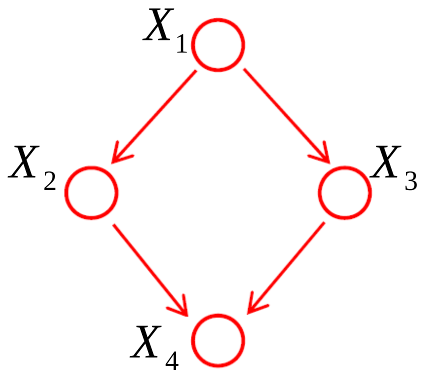


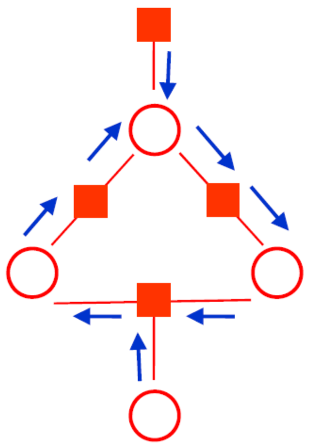


- Messages are merged, at this point we have to sum over several variables (execution time becomes non-linear).
- Basic idea carries over from inference on linear chain: sum out variables successively.
- Details in the Bishop textbook („Sum-Product“ algorithm)

Loopy Belief Propagation

- Inference in graphs that are not polytrees?
- Iterative message passing scheme, not exact anymore because of cycles in the graph (=approximate inference algorithm).





„Loopy Belief Propagation“

$$p(X_1, X_2, X_3, X_4) = p(X_1)p(X_2 \mid X_1)p(X_3 \mid X_1)p(X_4 \mid X_2, X_3)$$

- Exact inference in non-polytree graphs: Transform graph into equivalent acyclic graph („Junction Tree“ algorithm).