# 15-780 – Probabilistic Inference

J. Zico Kolter

March 30, 2014

# Outline

Probabilistic graphical models

Exact inference

Approximate inference

# Outline

Probabilistic graphical models

Exact inference

Approximate inference

# Probability distributions

- Probabilistic graphical models (PGMs) are about representing probability distributions over random variables

$$p(x)$$

where for this lecture, $x \in \{0,1\}^n$, $p : \{0,1\}^n \to [0,1]$

- Naively, since there are $2^n$ possible assignments to $x$, can represent this distribution completely using $2^n - 1$ numbers, but quickly becomes intractable for large $n$

- PGMs are methods to represent these distributions more compactly, by exploiting *conditional independence*

# Bayesian networks

- A Bayesian network is defined by:

  1. A directed acyclic graph (DAG) $G = (V = \{x_1, \ldots, x_n\}, E)$

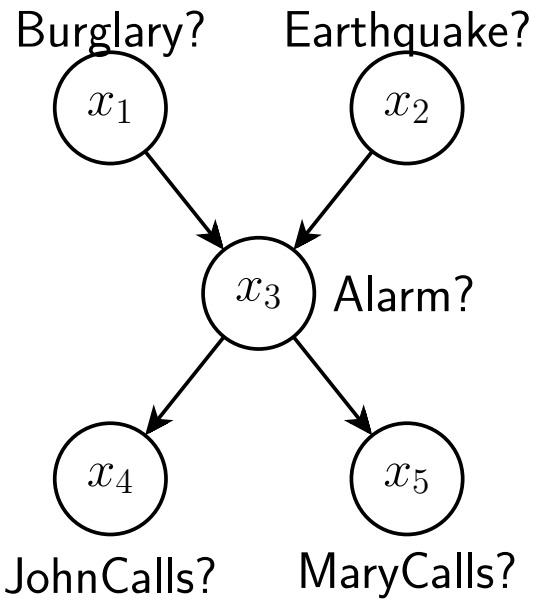
  2. A set of conditional probability tables $p(x_i|\mathrm{Parents}(x_i))$

- Defines the joint probability distribution

$$p(x) = \prod_{i=1}^{n} p(x_i|\mathrm{Parents}(x_i))$$

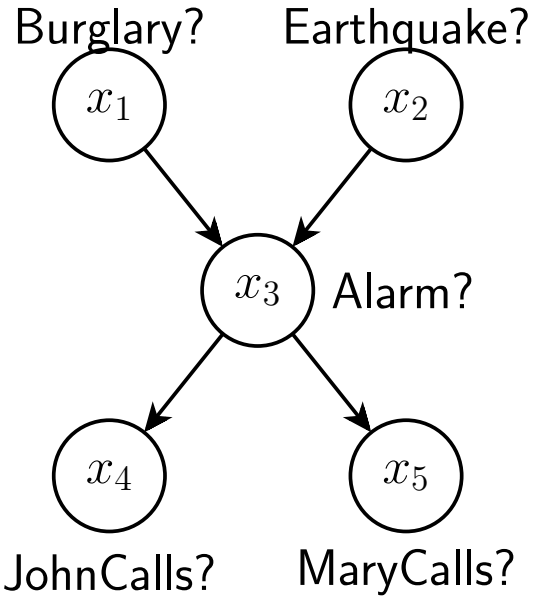
- Equivalently, each node is conditionally independent of all non-descendants given its parents

# Bayes net example



# Bayes net example

| $p(x_1 = 1)$ |
|---|
| 0.001 |




| $p(x_2 = 1)$ |
|---|
| 0.002 |

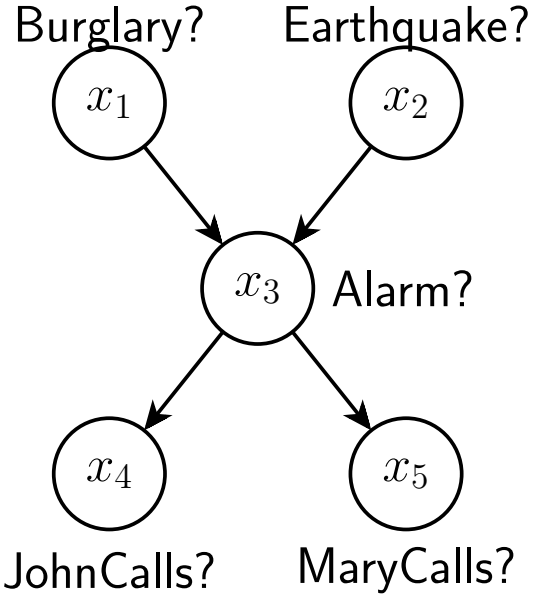
| $x_1$ | $x_2$ | $p(x_3 = 1)$ |
|---|---|---|
| 0 | 0 | 0.001 |
| 0 | 1 | 0.29 |
| 1 | 0 | 0.94 |
| 1 | 1 | 0.95 |

| $x_3$ | $p(x_4 = 1)$ |
|---|---|
| 0 | 0.05 |
| 1 | 0.9 |

| $x_3$ | $p(x_5 = 1)$ |
|---|---|
| 0 | 0.01 |
| 1 | 0.7 |

# Bayes net example

| $p(x_1 = 1)$ |
|---|
| 0.001 |




| $p(x_2 = 1)$ |
|---|
| 0.002 |

| $x_1$ | $x_2$ | $p(x_3 = 1)$ |
|---|---|---|
| 0 | 0 | 0.001 |
| 0 | 1 | 0.29 |
| 1 | 0 | 0.94 |
| 1 | 1 | 0.95 |

| $x_3$ | $p(x_4 = 1)$ |
|---|---|
| 0 | 0.05 |
| 1 | 0.9 |

| $x_3$ | $p(x_5 = 1)$ |
|---|---|
| 0 | 0.01 |
| 1 | 0.7 |

- Can write distribution as

$$p(x) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)p(x_4|x_1, x_2, x_3)p(x_5|x_1, x_2, x_3, x_4)$$
$$= p(x_1)p(x_2)p(x_3|x_1, x_2)p(x_4|x_3)p(x_5|x_3)$$

# Markov random fields

- A (pairwise) Markov random field (MRF) is defined by:

  1. An *undirected* graph $G = (V = \{x_1, \ldots, x_n\}, E)$

  2. A set of *unary* potential $f(x_i)$ for each $i = 1, \ldots, n$

  3. A set of *binary* potentials $f(x_i, x_j)$ for all $i, j \in E$

- Defines the joint probability distribution

$$p(x) = \frac{1}{Z} \prod_{i=1}^{n} f(x_i) \prod_{i,j \in E} f(x_i, x_j)$$

where $Z$ is a *normalization constant* (also called partition function)

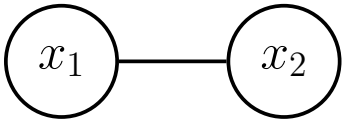
$$Z = \sum_{x} \prod_{i=1}^{n} f(x_i) \prod_{i,j \in E} f(x_i, x_j)$$

- Equivalently, each node is in MRF is conditionally independent of all other nodes given it's neighbors

$$p(x_i|x_{-i}) = p(x_i|\text{Neighbors}(x_i))$$

not trivial to show, known as Hammersley-Clifford theorem

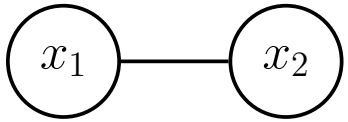
# MRF example



# MRF example

| $x_1$ | $x_2$ | $f(x_1, x_2)$ |
|---|---|---|
| 0 | 0 | 10 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 10 |

| $x_1$ | $f(x_1)$ |
|---|---|
| 0 | 1 |
| 1 | 5 |

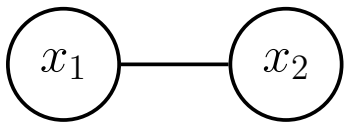



| $x_2$ | $f(x_1)$ |
|---|---|
| 0 | 5 |
| 1 | 1 |

# MRF example

| $x_1$ | $x_2$ | $f(x_1, x_2)$ |
|---|---|---|
| 0 | 0 | 10 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 10 |

| $x_1$ | $f(x_1)$ |
|---|---|
| 0 | 1 |
| 1 | 5 |




| $x_2$ | $f(x_1)$ |
|---|---|
| 0 | 5 |
| 1 | 1 |

| $x_1$ | $x_2$ | $\Pi f$ | $p(x)$ |
|---|---|---|---|
| 0 | 0 | 50 | 1/3 |
| 0 | 1 | 25 | 1/6 |
| 1 | 0 | 25 | 1/6 |
| 1 | 1 | 50 | 1/3 |

- E.g. $p(x_1 = 1, x_2 = 1) = \frac{1}{150} 5 \cdot 10 \cdot 1 = 1/3$

# Factor graphs

- A generalization that captures both Bayesian networks and Markov random fields

- An undirected graph, $G = \{V = \{x_1, \ldots, x_n, f_1, \ldots, f_m\}, E\}$ over variables and factors

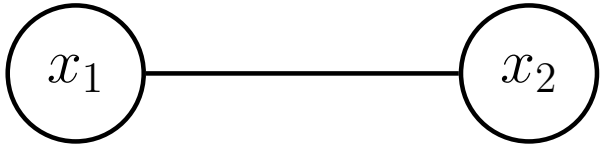
- There exists an edge $f_i$ — $x_j$ if and only if factor $f_i$ includes variable $x_j$

- Defines the joint probability distribution

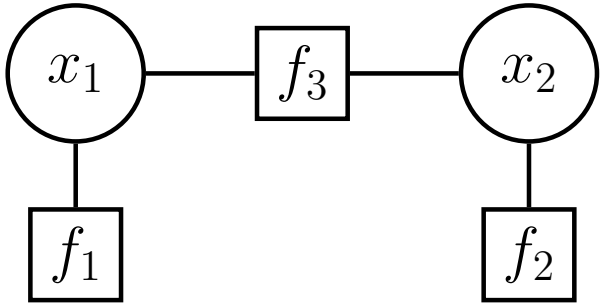
$$p(x) = \frac{1}{Z} \prod_{i=1}^{m} f_i(\mathcal{X}_i)$$

where $\mathcal{X}_i = \{x_j : (f_i, x_j) \in E\})$ are all variables in factor $f_i$

# MRF to factor graph

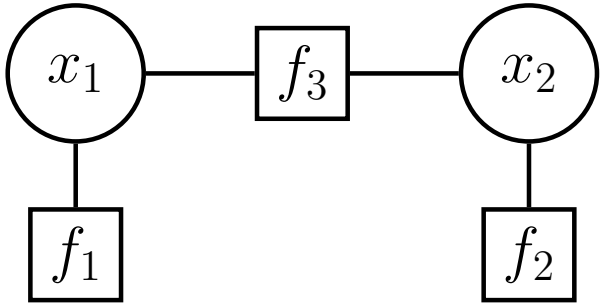


# MRF to factor graph

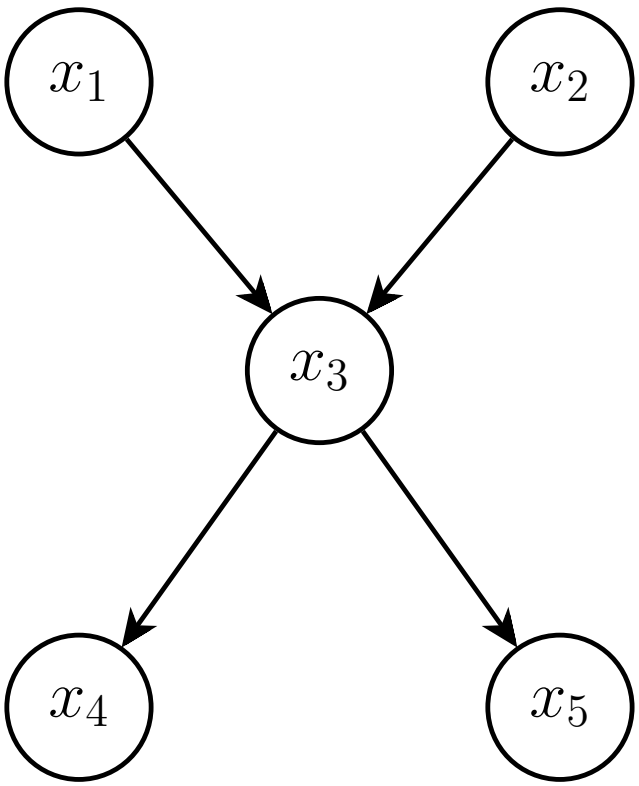


# MRF to factor graph

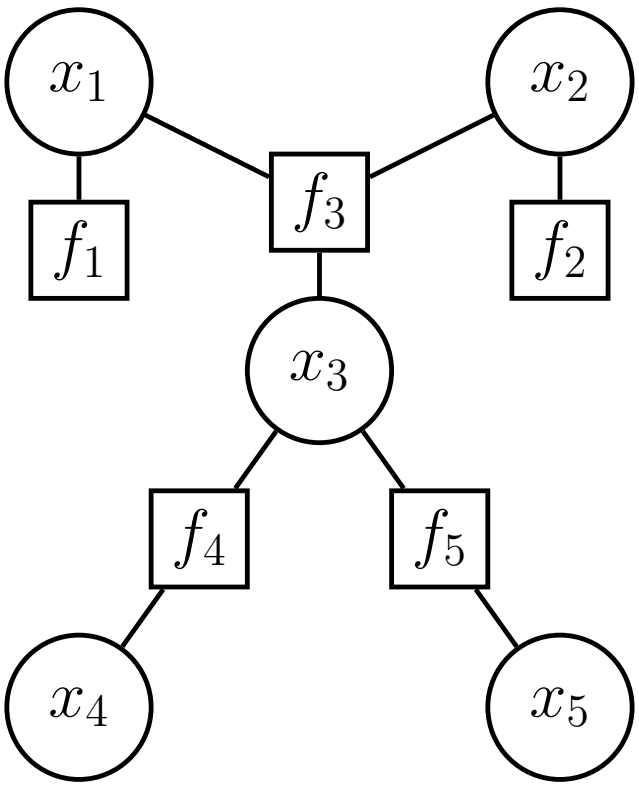
| $x_1$ | $x_2$ | $f_3(x_1, x_2)$ |
|---|---|---|
| 0 | 0 | 10 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 10 |



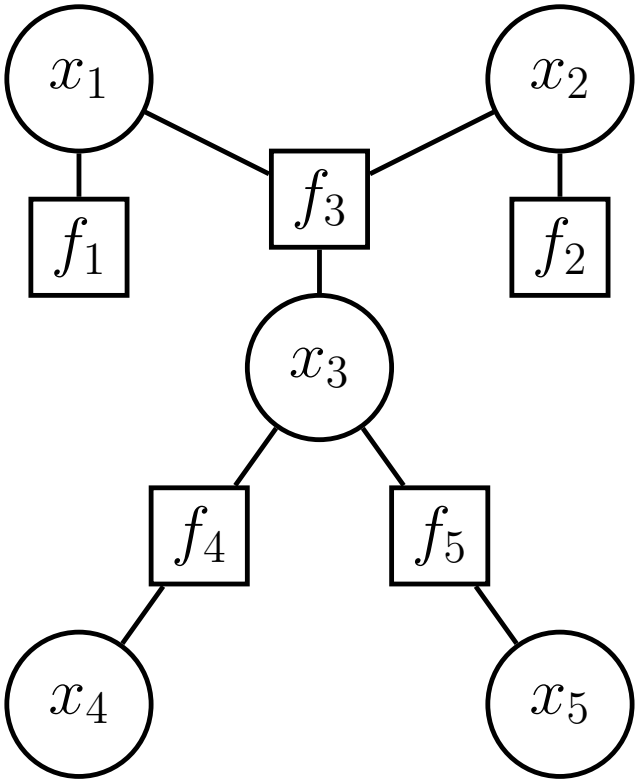
# Bayes net to factor graph



# Bayes net to factor graph

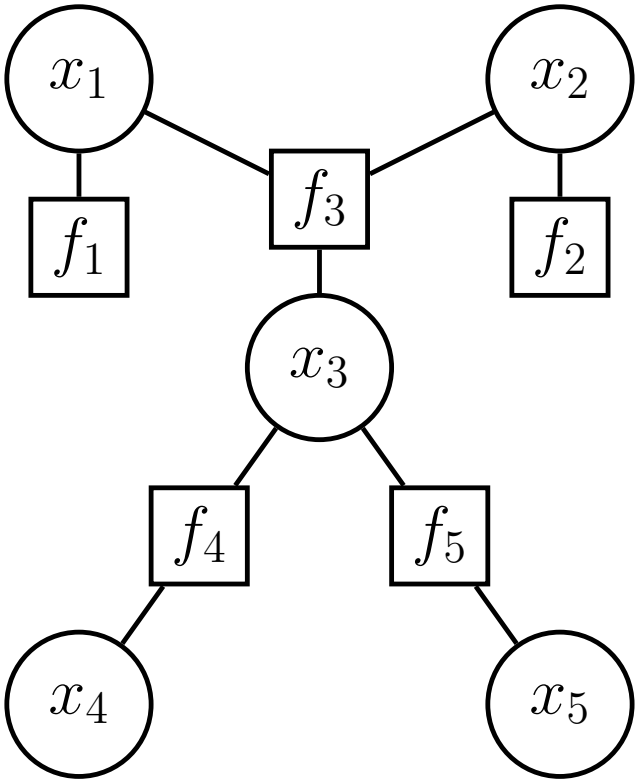


# Bayes net to factor graph




| $x_3$ | $p(x_5 = 1)$ |
|---|---|
| 0 | 0.01 |
| 1 | 0.7 |

# Bayes net to factor graph




| $x_3$ | $p(x_5 = 1)$ |
|---|---|
| 0 | 0.01 |
| 1 | 0.7 |

| $x_3$ | $x_5$ | $f_5(x_3, x_5)$ |
|---|---|---|
| 0 | 0 | 0.99 |
| 0 | 1 | 0.01 |
| 1 | 0 | 0.3 |
| 1 | 1 | 0.7 |

# Outline

Probabilistic graphical models

Exact inference

Approximate inference

# Inference in probabilistic graphical models

- *Inference* generally refers to methods that query probabilities given a graphical model

- Several types that come up frequently

  - *Marginal inference*: compute $p(x_{\mathcal{I}})$ for some $x_{\mathcal{I}} \subseteq \{x_1, \ldots, x_n\}$ (non-trivial even for $x_{\mathcal{I}} = \{x_1, \ldots, x_n\}$ in factor graph)

  - *Conditional inference*: compute $p(x_{\mathcal{I}} | x_{\mathcal{E}} = x_{\mathcal{E}}^0)$ for some $x_{\mathcal{I}}, x_{\mathcal{E}} \subseteq \{x_1, \ldots, x_n\}$, $x_{\mathcal{I}} \cap x_{\mathcal{E}} = \emptyset$

  - *Maximum a posteriori (MAP) inference*: compute $\max_{x_{\mathcal{I}}} p(x_{\mathcal{I}})$, and possibly the maximizing assignment $x_{\mathcal{I}}^\star$ (also, conditional analogue); also called most probable explanation (MPE)

# Inference via enumeration

- If we're willing to enumerate all $2^n$ possible values, inference queries can be answered easily
  - Marginal inference:

$$p(x_{\mathcal{I}}) = \sum_{\bar{x}_{\mathcal{I}}} p(x_{\mathcal{I}}, \bar{x}_{\mathcal{I}}) = \sum_{\bar{x}_{\mathcal{I}}} \prod_{i=1}^{m} f_i(\mathcal{X}_i)$$

  - Conditional inference

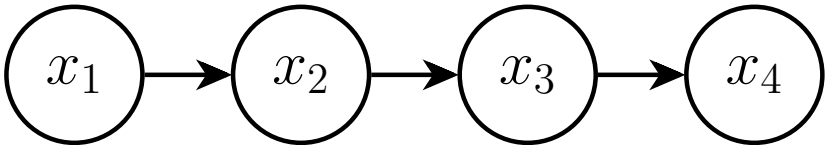
$$p(x_{\mathcal{I}} | x_{\mathcal{E}} = x_{\mathcal{E}}^0) = \frac{p(x_{\mathcal{I}}, x_{\mathcal{E}} = x_{\mathcal{E}}^0)}{p(x_{\mathcal{E}} = x_{\mathcal{E}}^0)}$$

  - MAP inference: compute $p(x_{\mathcal{I}} = x_{\mathcal{I}}^0)$ for all possible assignments $x_{\mathcal{I}}^0$, choose largest

# Exploiting graph structure in inference

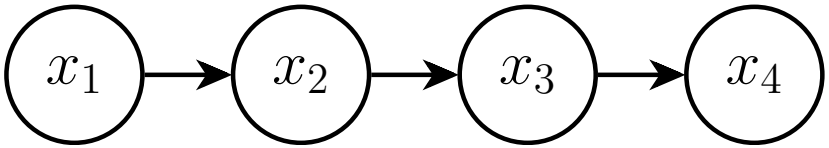
- When $n$ gets large, inference by exact enumeration is intractable
- Can (sometimes) use compact graph representation of the distribution to derive compact forms of inference

# Example: chain Bayesian network



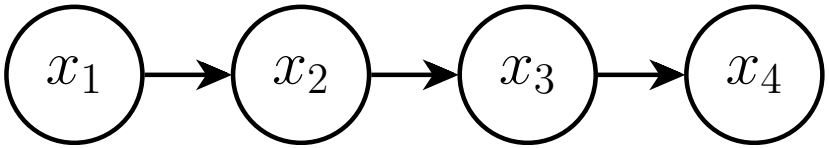

$$p(x_4) = \sum_{x_1, x_2, x_3} p(x_1, x_2, x_3, x_4)$$

# Example: chain Bayesian network



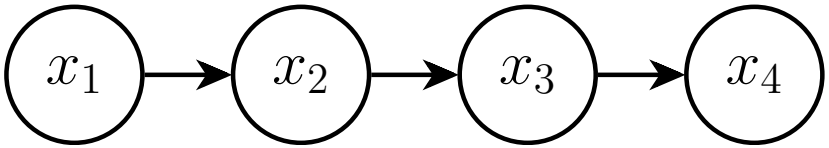

$$p(x_4) = \sum_{x_1, x_2, x_3} p(x_1) p(x_2|x_1) p(x_3|x_2) p(x_4|x_3)$$

# Example: chain Bayesian network



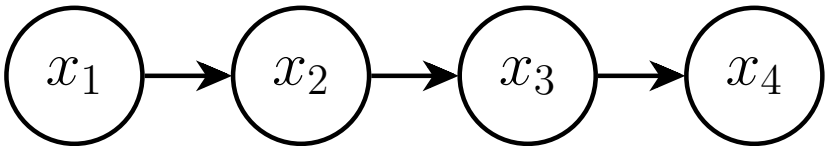

$$p(x_4) = \sum_{x_2,x_3} p(x_3|x_2)p(x_4|x_3) \sum_{x_1} p(x_1)p(x_2|x_1)$$

# Example: chain Bayesian network



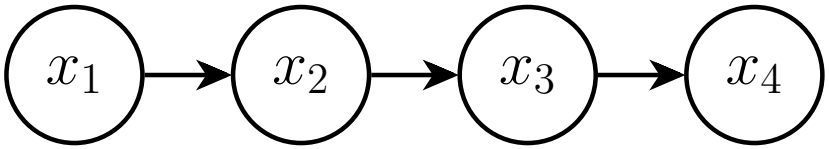

$$p(x_4) = \sum_{x_2, x_3} p(x_3|x_2)p(x_4|x_3)p(x_2)$$

# Example: chain Bayesian network




$$p(x_4) = \sum_{x_3} p(x_4|x_3) \sum_{x_2} p(x_3|x_2)p(x_2)$$

# Example: chain Bayesian network




$$p(x_4) = \sum_{x_3} p(x_4|x_3)p(x_3) = p(x_4)$$

# General algorithm: variable elimination

**function** $G' =$ Sum-Product-Eliminate$(G, x_i)$

$//$ eliminate variable $x_i$ from the factor graph G

$\mathcal{F} \leftarrow \{f_j \in V : (f_j, x_i) \in E\}$

$\tilde{\mathcal{X}} \leftarrow \{x_k : (f_j, x_k) \in E, f_j \in \mathcal{F}\} - \{x_i\}$

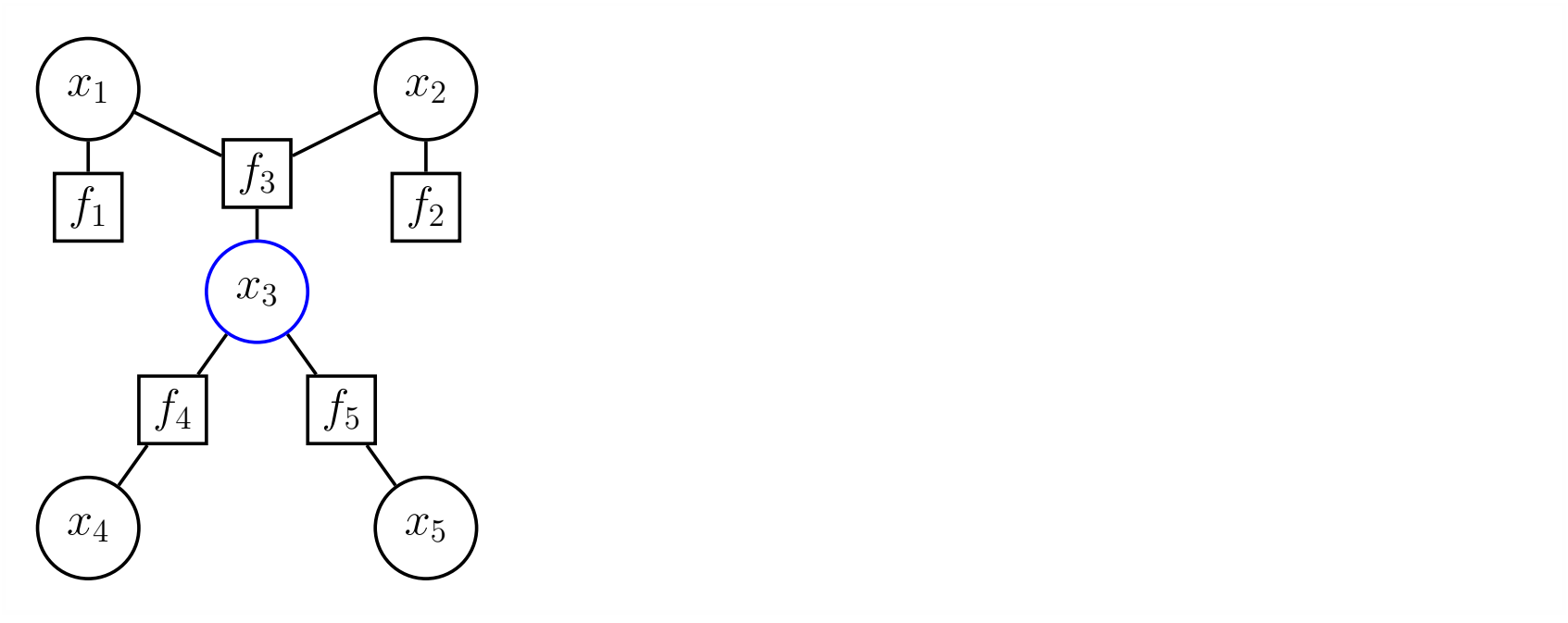
$\tilde{f}(\tilde{\mathcal{X}}) \leftarrow \sum_{x_i} \prod_{f_j \in \mathcal{F}} f_j(\mathcal{X}_j)$

$V' \leftarrow V - \{x_i, f_j \in \mathcal{F}\} + \{\tilde{f}\}$

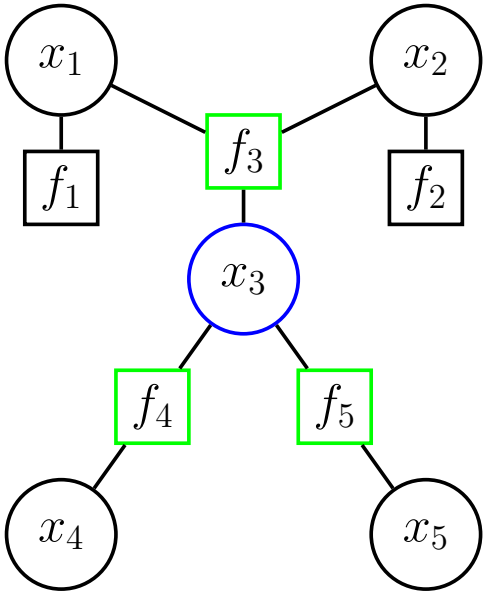
$E' \leftarrow E - \{(f_j, x_k) \in E : f_j \in \mathcal{F}\} + \{(\tilde{f}, x_k) : x_k \in \tilde{\mathcal{X}}\}$

**return** $G' = (V', E')$

# Variable elimination example

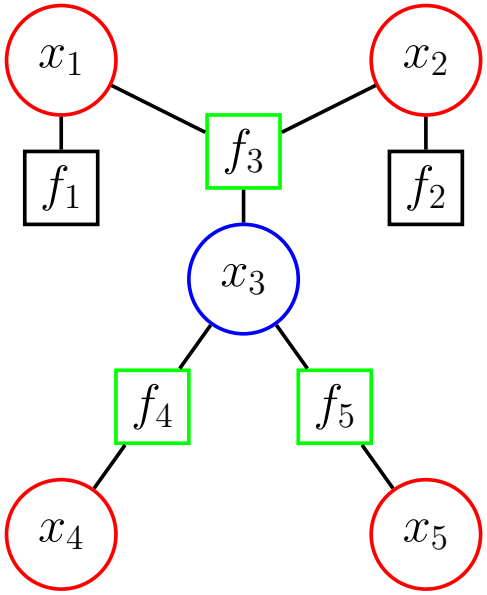


# Variable elimination example



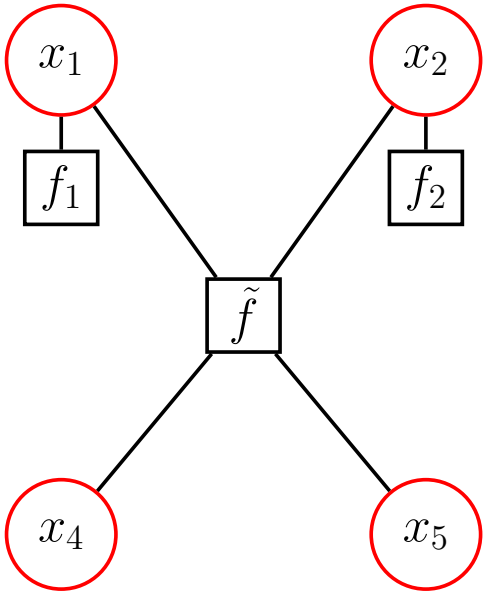

$\mathcal{F} = \{f_3, f_4, f_5\}$

# Variable elimination example




$$\mathcal{F} = \{f_3, f_4, f_5\}$$
$$\tilde{\mathcal{X}} = \{x_1, x_2, x_4, x_4\}$$

# Variable elimination example




$$\mathcal{F} = \{f_3, f_4, f_5\}$$
$$\tilde{\mathcal{X}} = \{x_1, x_2, x_4, x_5\}$$
$$\tilde{f}(x_1, x_2, x_3, x_5) = \sum_{x_3} f_3(x_1, x_2, x_3) f_4(x_3, x_4) f_5(x_3, x_5)$$
$$V' = \{x_1, x_2, x_4, x_5, f_1, f_2, \tilde{f}\}$$
$$E' = \{(f_1, x_1), (f_2, x_2), (\tilde{f}, x_1), (\tilde{f}, x_2), (\tilde{f}, x_4), (\tilde{f}, x_5)\}$$

- Full variable elimination algorithm just repeatedly eliminates variables

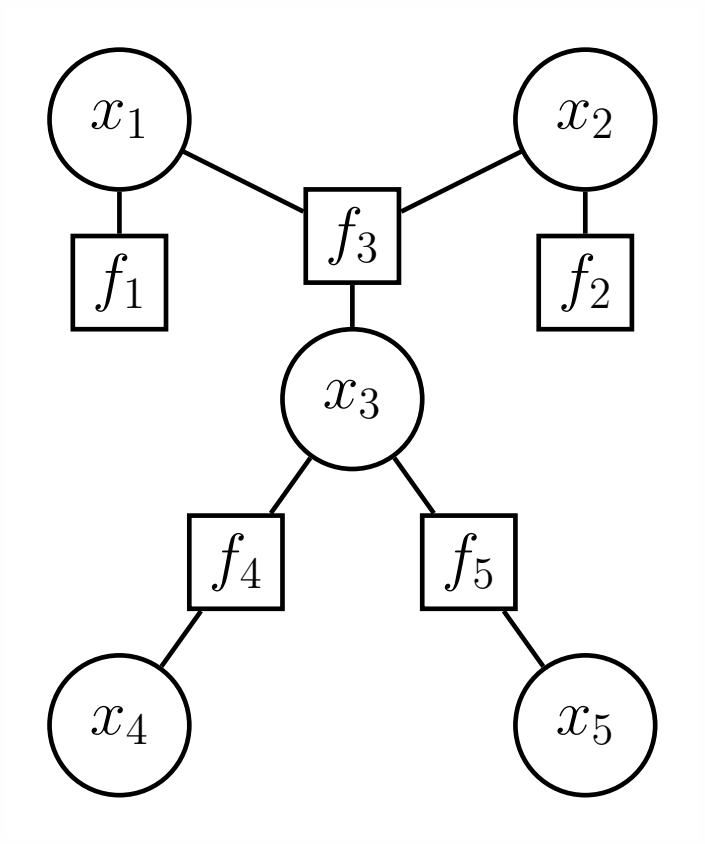
  **function** $G' =$ Sum-Product-Variable-Elimination$(G, X)$
  
  $\quad$ // eliminate an ordered list of variables $X$
  
  $\quad$ **for** $x_i \in X$ :
  
  $\quad\quad$ $G \leftarrow$ Sum-Product-Eliminate$(G, x_i)$
  
  $\quad$ **return** $G$

- Graph returned at the end is a marginalized factor graph over non-eliminated variables (eliminating all variables returns constant equal to partition function $Z$)

- The ordering matters a lot, eliminating variables in the wrong order can make algorithm no better than enumeration

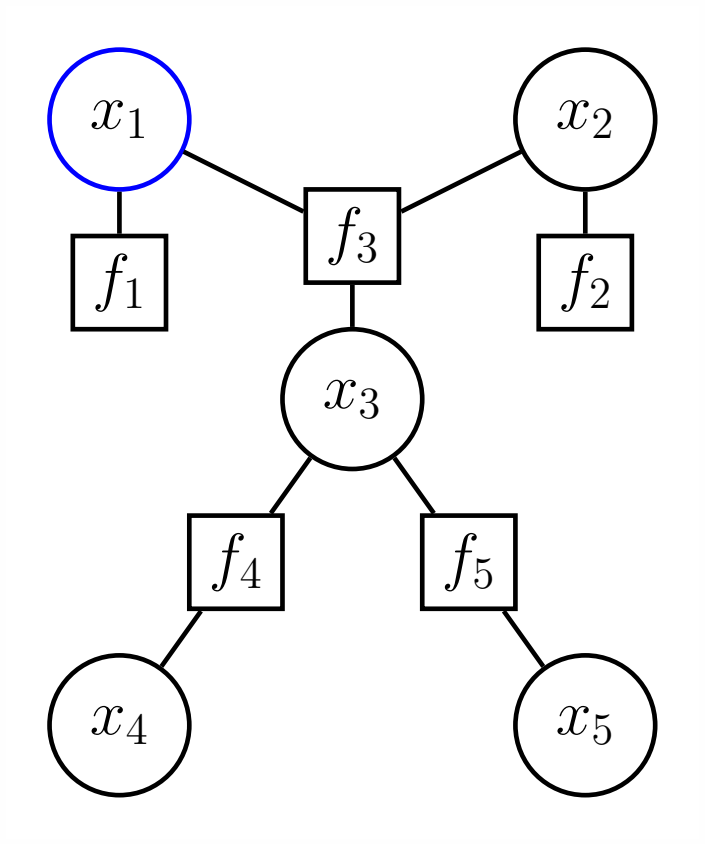
# Variable elimination example

Goal: compute $p(x_4)$



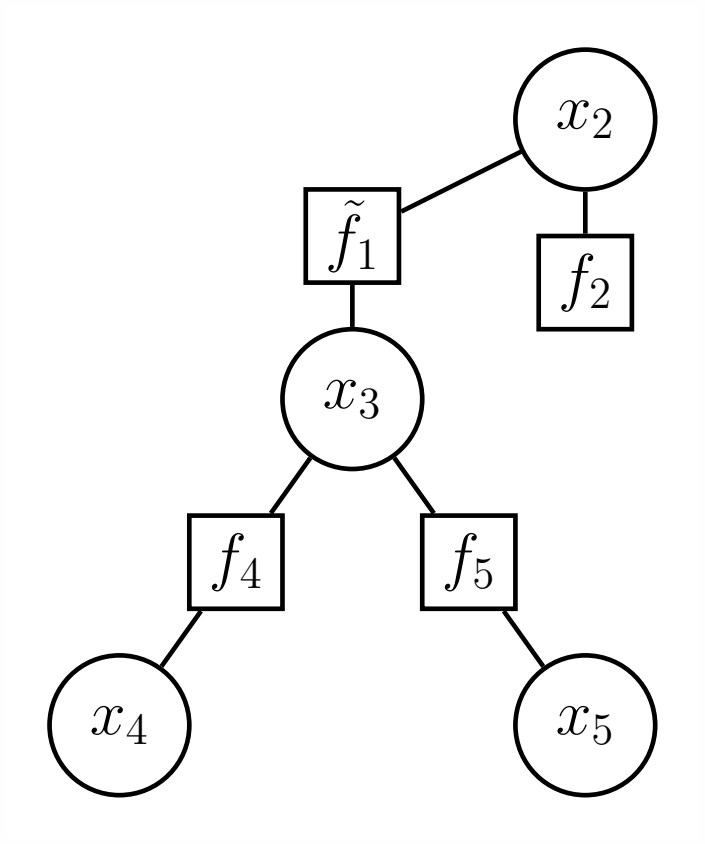
# Variable elimination example

Goal: compute $p(x_4)$



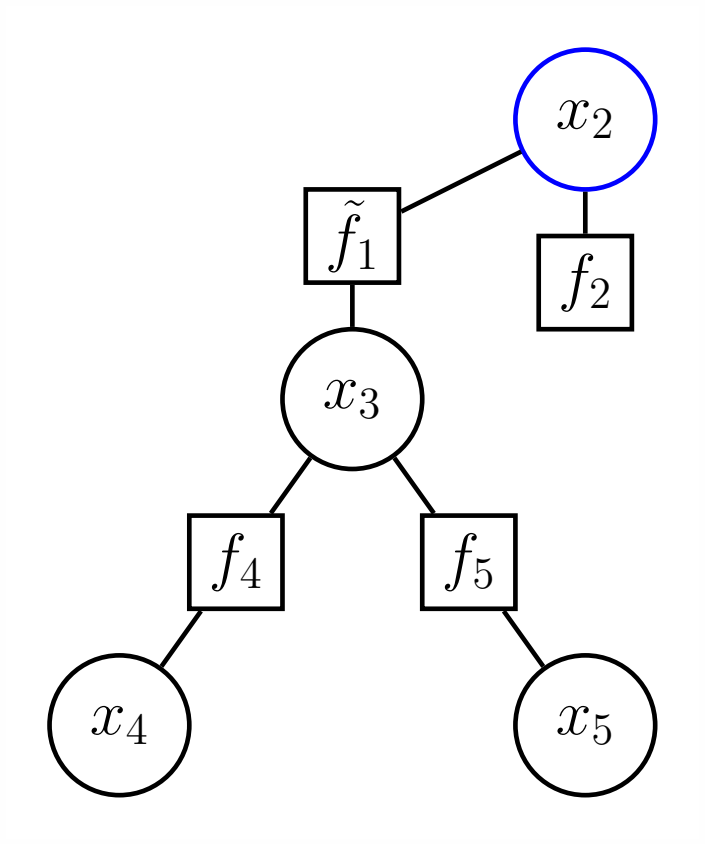
# Variable elimination example

Goal: compute $p(x_4)$



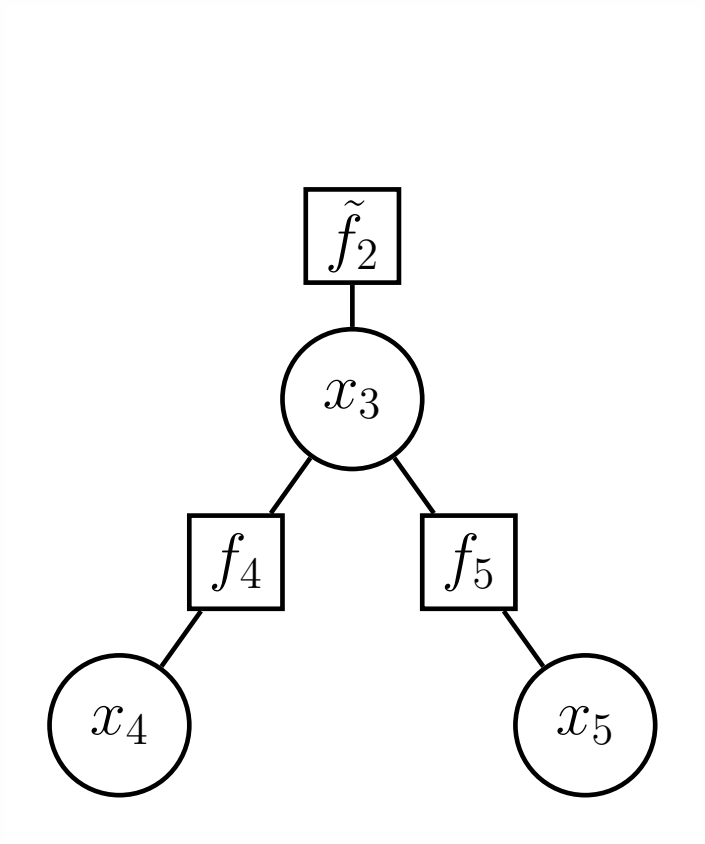
# Variable elimination example

Goal: compute $p(x_4)$



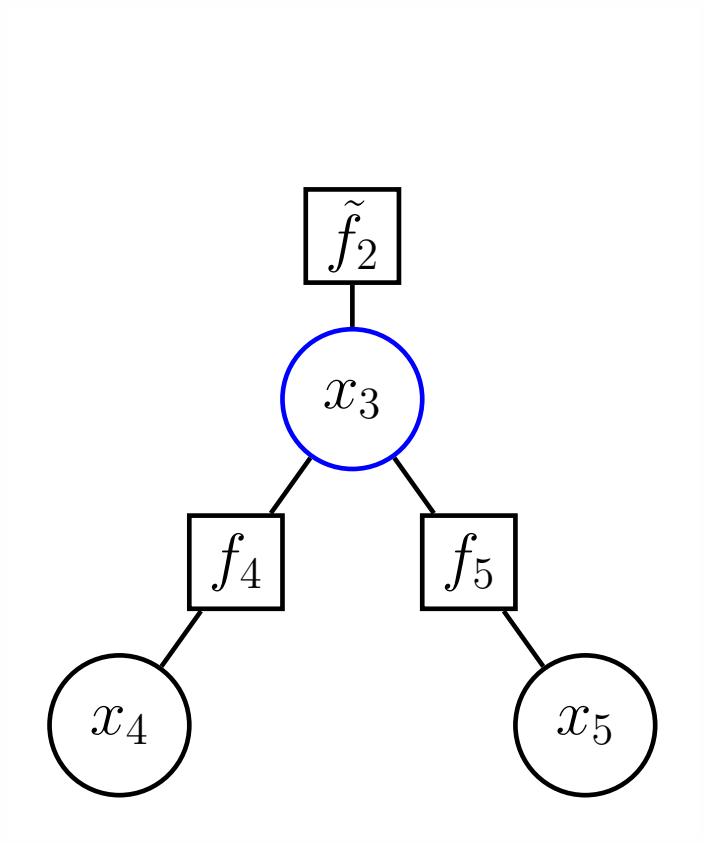
# Variable elimination example

Goal: compute $p(x_4)$



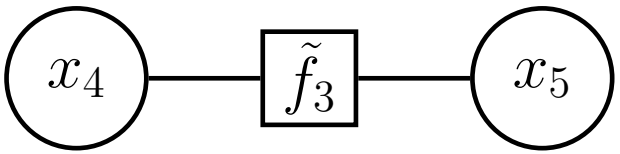
# Variable elimination example

Goal: compute $p(x_4)$



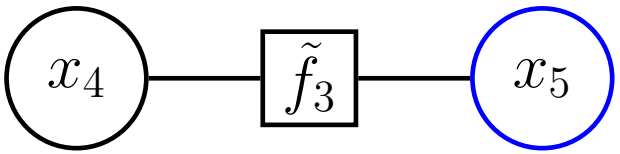
# Variable elimination example

Goal: compute $p(x_4)$



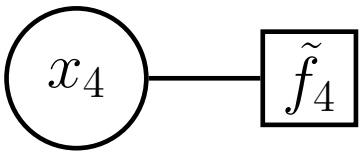
# Variable elimination example

Goal: compute $p(x_4)$



# Variable elimination example

Goal: compute $p(x_4)$



# Pitfalls

- *Tree-width* of graphical model is size of the maximum factor formed during variable elimination (assuming best ordering); inference is exponential in tree width

- But...
  - Finding best variable elimination ordering is NP-hard
  - Some "simple" graphs have high tree-width (e.g., $M \times N$ "grid" MRF has tree-width $\min(M, N)$)

# Extensions

- Difficulty with variable elimination as stated is that we need to "rerun" algorithm each time we want to make an inference query

- Solution: slight extension of variable elimination that caches intermediate factors, making a forward and backward pass over all variables (Junction Tree or Clique Tree algorithm)

- You'll probably see these algorithms written in terms of message passing, but these "messages" are just intermediate factors $\tilde{f}$'s

# MAP Inference

- Virtually identical approach can be applied to MAP inference

- Only change is replacing sum-product operation

$$\tilde{f}(\tilde{\mathcal{X}}) \leftarrow \sum_{x_i} \prod_{f_j \in \mathcal{F}} f_j(\mathcal{X}_j)$$

  with max-product operation

$$\tilde{f}(\tilde{\mathcal{X}}) \leftarrow \max_{x_i} \prod_{f_j \in \mathcal{F}} f_j(\mathcal{X}_j)$$

- If we want to find actual maximizing assignment, also need to keep a separate function of which $x_i$ value is maximal for each $\tilde{f}(\mathcal{X})$

# Outline

Probabilistic graphical models

Exact inference

Approximate inference

# Sampling methods

- Instead of exactly computing probabilities $p(x)$, we may want to draw random samples from this distribution $x \sim p(x)$

- For example, in Bayesian networks this is straightforward, just sample individual variables sequentially

$$x_i \sim p(x_i | \text{Parents}(x_i)) \;\; i = 1, \ldots, n$$

- For cases where we can efficiently perform variable elimination, a slightly modified procedure lets us draw random samples (perhaps conditioned on evidence)

# Gibbs sampling

- But what about cases too big for variable elimination?

- A common solution: Gibbs sampling

**function** $x =$ Gibbs-Sampling$(G, x, K)$
  **for** $i = 1, \ldots, K$:
    Choose a random $x_i$
    Sample $x_i \sim p(x_i | x_{-i}) \propto \prod_{f_j : (f_j, x_i) \in G} f_j(\mathcal{X}_j)$

- In the limit, $x$ will be drawn exactly according to the desired distribution (but may take exponentially long to converge)

- One of a broad class of methods called Markov Chain Monte Carlo (MCMC)

# Inference as optimization

- Inference in graphical models can be cast as an optimization problem, has been a huge source of ideas for improving exact and approximate inference methods

- We're going to consider the simpler case of MAP inference, which already looks like an optimization problem

$$\underset{x}{\text{maximize}}\, p(x)$$

- To put this is a form that we're more familiar with, for each factor $f_i$ define the optimization variable $\mu_i \in \mathbb{R}^{2^{|\mathcal{X}_i|}}$; $\mu_i$ should be thought of an an indicator for the assignment to $\mathcal{X}_i$

- Abusing notation a bit, we can write optimization as a binary integer program

$$\begin{aligned} \underset{\mu_1,\ldots,\mu_m}{\text{maximize}} \quad & \log p(\mu) = \sum_{i=1}^{m} \mu_i^T (\log f_i) \\ \text{subject to} \quad & \mu_1, \ldots, \mu_n \text{ is valid distribution} \\ & (\mu_i)_j \in \{0,1\}, \quad \forall i, j \end{aligned}$$

- "Valid distribution" here means assignments have to be consistent, i.e., if $x_k \in \mathcal{X}_i$ and $x_k \in \mathcal{X}_j$, then

$$\sum_{\mathcal{X}_i - \{x_k\}} \mu_i(\mathcal{X}_i) = \sum_{\mathcal{X}_i - \{x_k\}} \mu_j(\mathcal{X}_j)$$

and they have to have only one non-zero entry $\sum_j (\mu_i)_j = 1$

- This is still a hard, binary integer programming task, but it turns out that the LP relaxation is sometimes tight (i.e., just removing the integer constraints still gives the optimal solution)

- One case where relaxation is tight: tree factor graphs (these are ones we could already solve with max-product)

- Extremely cool: there are other cases where relaxation is *still* tight even though naive max-product doesn't apply, like certain grid MRFs

- Can also apply to the case of marginal inference (let $\mu$ terms have non-integer values, but also include terms due to partition function, other constraints)

- A big area of open research

# Take home points

- Probabilistic models can compactly represent high dimensional probability distribution
- Inference algorithms provide a method for making probabilistic queries that also (try to) exploit the structure of the distribution
- Wide range of inference methods, ranging from variable elimination for exact inference, sampling and optimization approaches