Inference and Representation

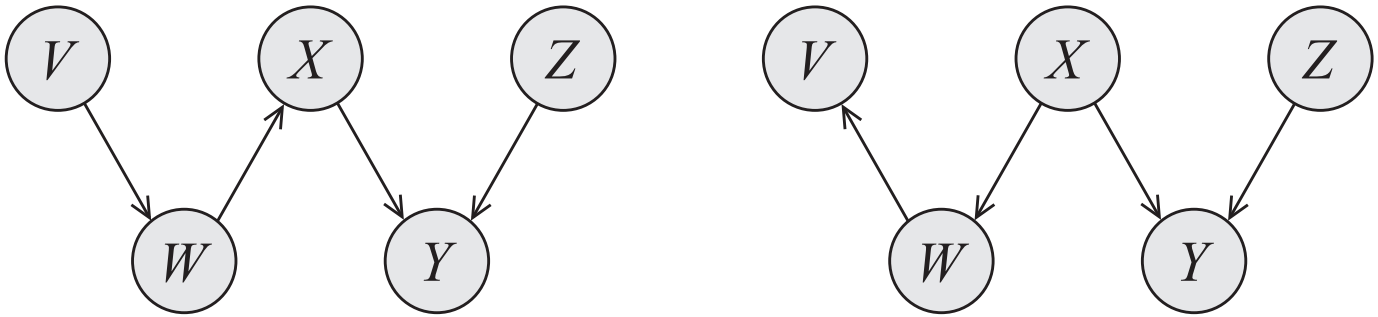
David Sontag

New York University

Lecture 6, Oct. 7, 2014

V-structures

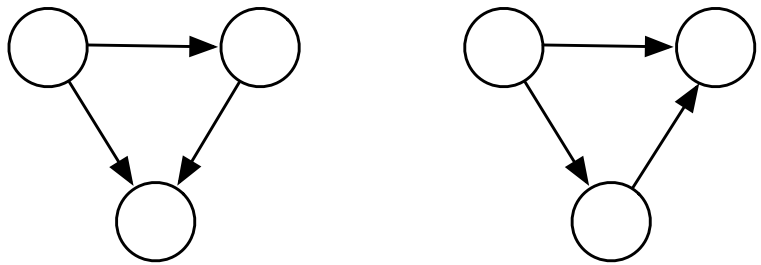
- **Theorem 3.7 (K&F)**: If $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same v-structures and the same undirected skeleton, then $I(\mathcal{G}_1) = I(\mathcal{G}_2)$



- Other direction does not apply! Why not?

V-structures

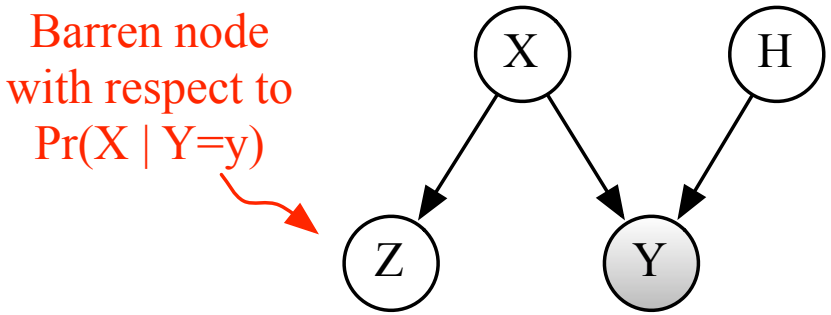
- **Theorem 3.7 (K&F)**: If $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same V-structures and the same undirected skeleton, then $I(\mathcal{G}_1) = I(\mathcal{G}_2)$.
- Other direction does not apply! Consider, for example, a complete graph:



- **Def:** A v-structure $X \rightarrow Z \leftarrow Y$ is an *immorality* if there is no directed edge between X and Y
- **Theorem 3.8 (K&F)**: $\mathcal{G}_1$ and $\mathcal{G}_2$ are I-equivalent if and only if $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same skeleton and the same set of immoralities.

Pruning nodes in Bayesian networks

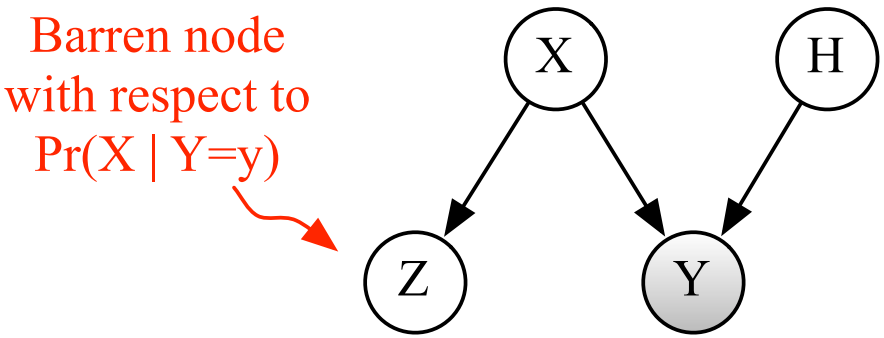
- A node in a Bayesian network $\mathcal{G}$ is a *leaf* if it has no children.
- **Def:** A node is *barren* w.r.t. a query $p_{\mathcal{G}}(\mathbf{X} \mid \mathbf{Y} = \mathbf{y})$ if it is a leaf and it is not in $\mathbf{X} \cup \mathbf{Y}$



- To *remove* a node v from a Bayesian network $\mathcal{G} = (V, E)$ means:
  1. Removing v from V, and removing from E all edges to/from v
  2. Leave the CPDs for the rest of the variables the same
- **Theorem:** Let $\mathcal{G}'$ be the Bayesian network obtained from $\mathcal{G}$ by removing v. If v is barren w.r.t. the query $p_{\mathcal{G}}(\mathbf{X} \mid \mathbf{Y} = \mathbf{y})$, then

$$p_{\mathcal{G}}(\mathbf{X} \mid \mathbf{Y} = \mathbf{y}) = p_{\mathcal{G}'}(\mathbf{X} \mid \mathbf{Y} = \mathbf{y}).$$

Pruning nodes in Bayesian networks



$$
\begin{aligned}
p_{\mathcal{G}}(X = x \mid Y = y) &= \frac{\sum_{h,z} p_{\mathcal{G}}(z, x, y, h)}{\sum_{\hat{x},h,z} p_{\mathcal{G}}(z, \hat{x}, y, h)} \\
&= \frac{\sum_{h,z} \theta_x \theta_h \theta_{z|x} \theta_{y|x,h}}{\sum_{\hat{x},h,z} \theta_{\hat{x}} \theta_h \theta_{z|\hat{x}} \theta_{y|\hat{x},h}} \\
&= \frac{\sum_{h} \theta_x \theta_h \theta_{y|x,h} \sum_{z} \theta_{z|x}}{\sum_{\hat{x},h} \theta_{\hat{x}} \theta_h \theta_{y|\hat{x},h} \sum_{z} \theta_{z|\hat{x}}} \\
&= p_{\mathcal{G}'}(X = x \mid Y = y),
\end{aligned}
$$

where $\mathcal{G}'$ is the Bayesian network with Z removed.

Pruning nodes in Bayesian networks

- **Def:** A node is *barren* w.r.t. a query $p_{\mathcal{G}}(\mathbf{X} \mid \mathbf{Y} = \mathbf{y})$ if it is a leaf and it is not in $\mathbf{X} \cup \mathbf{Y}$
- **Theorem:** Let $\mathcal{G}'$ be the Bayesian network obtained from $\mathcal{G}$ by removing v. If v is barren w.r.t. the query $p_{\mathcal{G}}(\mathbf{X} \mid \mathbf{Y} = \mathbf{y})$, then

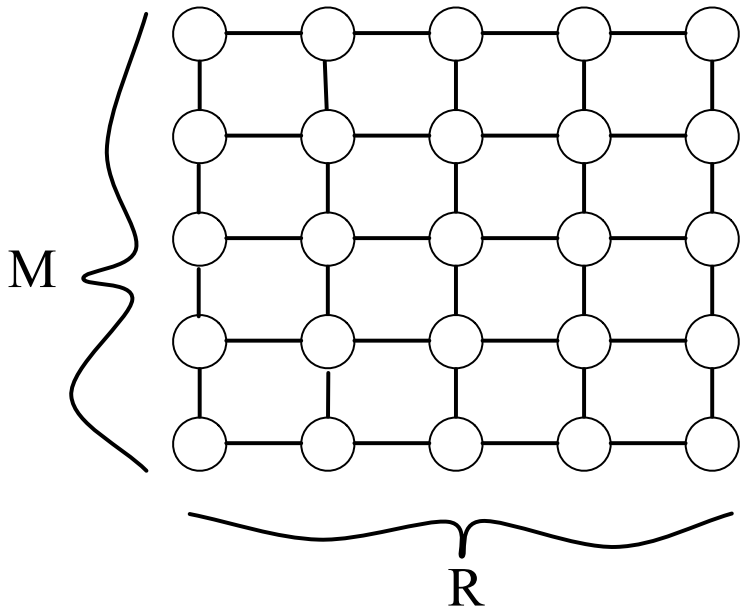
$$p_{\mathcal{G}}(\mathbf{X} \mid \mathbf{Y} = \mathbf{y}) = p_{\mathcal{G}'}(\mathbf{X} \mid \mathbf{Y} = \mathbf{y}).$$

- Let $An(\mathbf{X} \cup \mathbf{Y})$ be the *ancestral set* of $\mathbf{X} \cup \mathbf{Y}$, i.e. the set including $\mathbf{X} \cup \mathbf{Y}$ and all of their ancestors
- **Corollary:** All the nodes outside of $An(\mathbf{X} \cup \mathbf{Y})$ are irrelevant to the query $p_{\mathcal{G}}(\mathbf{X} \mid \mathbf{Y} = \mathbf{y})$ and can be removed
- **Theorem:** Let $\mathcal{G}'$ be the Bayesian network obtained from $\mathcal{G}$ by removing all nodes that are d-separated from X by Y. Then

$$p_{\mathcal{G}}(\mathbf{X} \mid \mathbf{Y} = \mathbf{y}) = p_{\mathcal{G}'}(\mathbf{X} \mid \mathbf{Y} = \mathbf{y}).$$

Treewidth of grid graph?

- What is the treewidth of a $M \times R$ node grid-structured MRF?



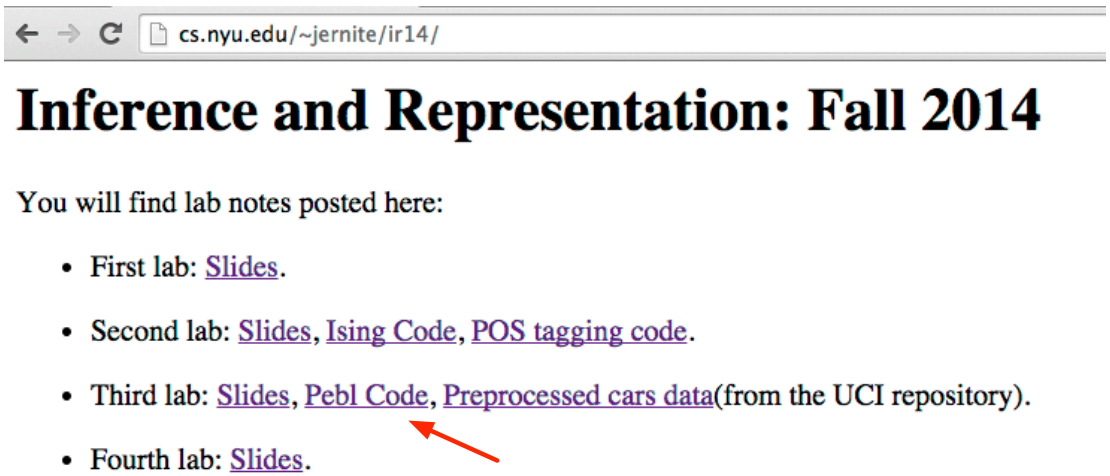
- Choose smaller size, and eliminate variables on that side in any order
- Will only introduce edges along the next row (column)
- Thus, treewidth is at most $min(M, R)$

Midterm exam, etc.

- **Make sure you are on the course mailing list**
- No class next week, 10/14 (fall recess)
- I *will* hold office hours 10/14 and there *will* be lab 10/16. Bring your midterm-related questions!
- Midterm exam in class on 10/21 (in 2 weeks). Closed book; no calculators/phones/computers
- Midterm covers everything up to and including this week's lab (10/9)
- PS4 released *after* midterm exam

How to study for the midterm

1. Go over slides and your notes from lecture & lab
2. Carefully go over required readings and the homeworks
3. Do practice problems from the end of the Murphy chapters, e.g. all of the questions for Chapters 10 and 20, and exercices 19.5 and 22.4
4. Experiment with structure learning code from 3rd lab:



Today's lecture

1. Integer linear programming
2. MAP inference as an integer linear program
3. Linear programming relaxations for MAP inference
4. Dual decomposition

MAP inference

- Recall the MAP inference task,

$$\arg\max_{\mathbf{x}} p(\mathbf{x}), \qquad p(\mathbf{x}) = \frac{1}{Z}\prod_{c\in C}\phi_c(\mathbf{x}_c)$$

(we assume any evidence has been subsumed into the potentials, as discussed in the last lecture)

- Since the normalization term is simply a constant, this is equivalent to

$$\arg\max_{\mathbf{x}} \prod_{c\in C}\phi_c(\mathbf{x}_c)$$

(called the *max-product* inference task)

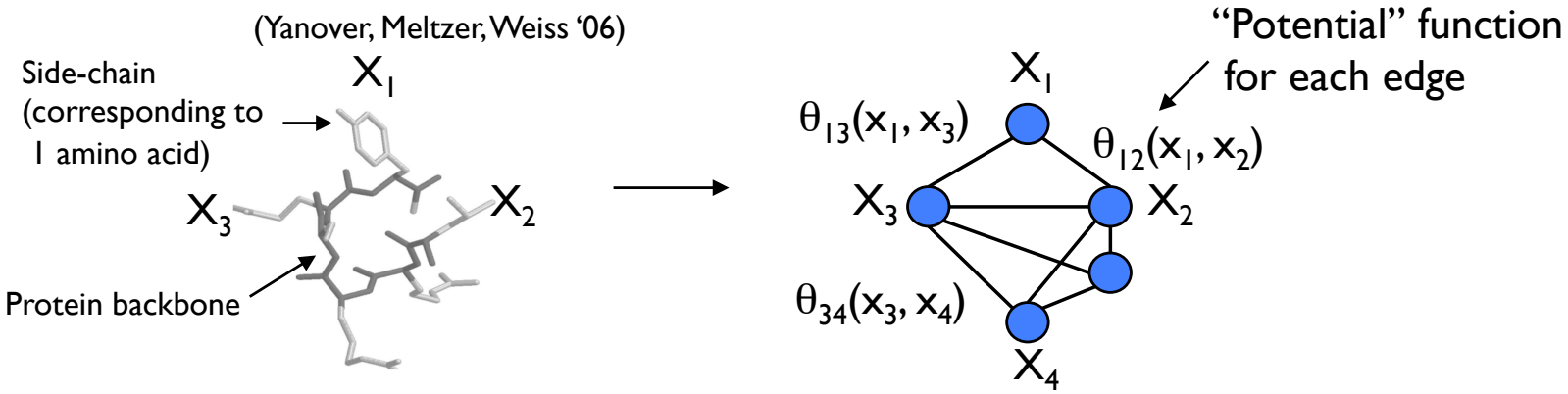
- Furthermore, since log is monotonic, letting $\theta_c(\mathbf{x_c}) = \lg \phi_c(\mathbf{x_c})$, we have that this is equivalent to

$$\arg\max_{\mathbf{x}} \sum_{c\in C}\theta_c(\mathbf{x}_c)$$

(called *max-sum*)

Motivating application: protein side-chain placement

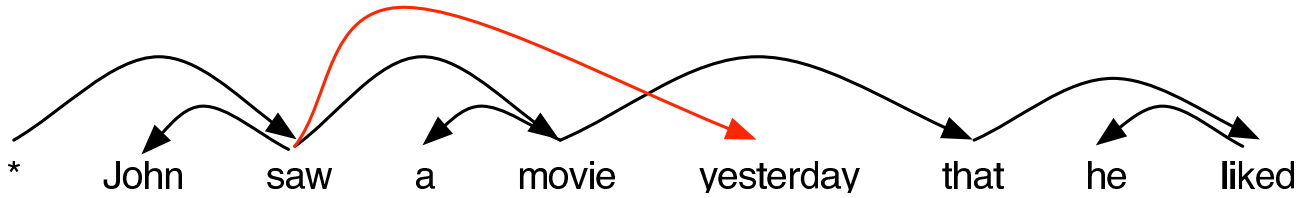
- Find "minimum energy" conformation of amino acid side-chains along a fixed carbon backbone:



- Orientations of the side-chains are represented by discretized angles called rotamers
- Rotamer choices for nearby amino acids are energetically coupled (attractive and repulsive forces)

Motivating application: dependency parsing

- Given a sentence, predict the dependency tree that relates the words:



- Arc from head word of each phrase to words that modify it
- May be *non-projective*: each word and its descendents may not be a contiguous subsequence
- m words $\Longrightarrow$ $m(m-1)$ binary arc selection variables $x_{ij} \in \{0,1\}$
- Let $\mathbf{x}_{|i} = \{x_{ij}\}_{j \neq i}$ (all outgoing edges). Predict with:

$$\max_{\mathbf{x}} \theta_T(\mathbf{x}) + \sum_{ij} \theta_{ij}(x_{ij}) + \sum_{i} \theta_{i|}(\mathbf{x}_{|i})$$

MAP as an integer linear program (ILP)

- MAP as a discrete optimization problem is

$$\arg\max_{\mathbf{x}} \sum_{i \in V} \theta_i(x_i) + \sum_{ij \in E} \theta_{ij}(x_i, x_j).$$

- To turn this into an integer linear program, we introduce indicator variables
  1. $\mu_i(x_i)$, one for each $i \in V$ and state x_i
  2. $\mu_{ij}(x_i, x_j)$, one for each edge $ij \in E$ and pair of states x_i, x_j
- The objective function is then

$$\max_{\mu} \sum_{i \in V} \sum_{x_i} \theta_i(x_i)\mu_i(x_i) + \sum_{ij \in E} \sum_{x_i, x_j} \theta_{ij}(x_i, x_j)\mu_{ij}(x_i, x_j)$$

- What is the dimension of μ, if binary variables?

What are the constraints?

- Force every "cluster" of variables to choose a local assignment:

$$\begin{aligned} \mu_i(x_i) &\in \{0,1\} \quad \forall i \in V, x_i \\ \sum_{x_i} \mu_i(x_i) &= 1 \quad \forall i \in V \\ \mu_{ij}(x_i, x_j) &\in \{0,1\} \quad \forall ij \in E, x_i, x_j \\ \sum_{x_i, x_j} \mu_{ij}(x_i, x_j) &= 1 \quad \forall ij \in E \end{aligned}$$

- Enforce that these local assignments are globally consistent:

$$\begin{aligned} \mu_i(x_i) &= \sum_{x_j} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_i \\ \mu_j(x_j) &= \sum_{x_i} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_j \end{aligned}$$

MAP as an integer linear program (ILP)

$$\text{MAP}(\theta) = \max_{\mu} \sum_{i \in V} \sum_{x_i} \theta_i(x_i)\mu_i(x_i) + \sum_{ij \in E} \sum_{x_i, x_j} \theta_{ij}(x_i, x_j)\mu_{ij}(x_i, x_j)$$

subject to:

$$\begin{aligned}
\mu_i(x_i) &\in \{0,1\} \quad \forall i \in V, x_i \\
\sum_{x_i} \mu_i(x_i) &= 1 \quad \forall i \in V \\
\mu_i(x_i) &= \sum_{x_j} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_i \\
\mu_j(x_j) &= \sum_{x_i} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_j
\end{aligned}$$

- Many extremely good off-the-shelf solvers, such as CPLEX and Gurobi

Linear programming relaxation for MAP

Integer linear program was:

$$\text{MAP}(\theta) = \max_{\mu} \sum_{i \in V} \sum_{x_i} \theta_i(x_i)\mu_i(x_i) + \sum_{ij \in E} \sum_{x_i, x_j} \theta_{ij}(x_i, x_j)\mu_{ij}(x_i, x_j)$$

subject to

$$\begin{aligned}
\mu_i(x_i) &\in \{0,1\} \quad \forall i \in V, x_i \\
\sum_{x_i} \mu_i(x_i) &= 1 \quad \forall i \in V \\
\mu_i(x_i) &= \sum_{x_j} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_i \\
\mu_j(x_j) &= \sum_{x_i} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_j
\end{aligned}$$

Relax integrality constraints, allowing the variables to be **between** 0 and 1:

$$\mu_i(x_i) \in [0,1] \quad \forall i \in V, x_i$$

Linear programming relaxation for MAP

Linear programming relaxation is:

$$\mathrm{LP}(\theta) = \max_{\mu} \sum_{i \in V} \sum_{x_i} \theta_i(x_i)\mu_i(x_i) + \sum_{ij \in E} \sum_{x_i, x_j} \theta_{ij}(x_i, x_j)\mu_{ij}(x_i, x_j)$$

$$\begin{aligned}
\mu_i(x_i) &\in [0,1] \quad \forall i \in V, x_i \\
\sum_{x_i} \mu_i(x_i) &= 1 \quad \forall i \in V \\
\mu_i(x_i) &= \sum_{x_j} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_i \\
\mu_j(x_j) &= \sum_{x_i} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_j
\end{aligned}$$

- Linear programs can be solved **efficiently**! Simplex method, interior point, ellipsoid algorithm
- Since the LP relaxation maximizes over a **larger** set of solutions, its value can only be *higher*

$$\mathrm{MAP}(\theta) \leq \mathrm{LP}(\theta)$$

- LP relaxation is **tight** for tree-structured MRFs

Today's lecture

1. Integer linear programming
2. MAP inference as an integer linear program
3. Linear programming relaxations for MAP inference
4. Dual decomposition

Dual decomposition

- Consider the MAP problem for pairwise Markov random fields:

$$\text{MAP}(\theta) = \max_{\mathbf{x}} \sum_{i \in V} \theta_i(x_i) + \sum_{ij \in E} \theta_{ij}(x_i, x_j).$$

- If we push the maximizations *inside* the sums, the value can only *increase*:

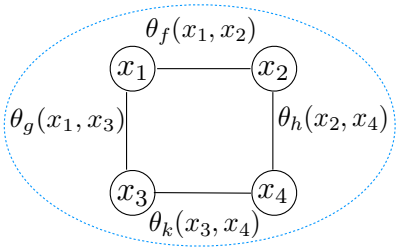
$$\text{MAP}(\theta) \leq \sum_{i \in V} \max_{x_i} \theta_i(x_i) + \sum_{ij \in E} \max_{x_i, x_j} \theta_{ij}(x_i, x_j)$$

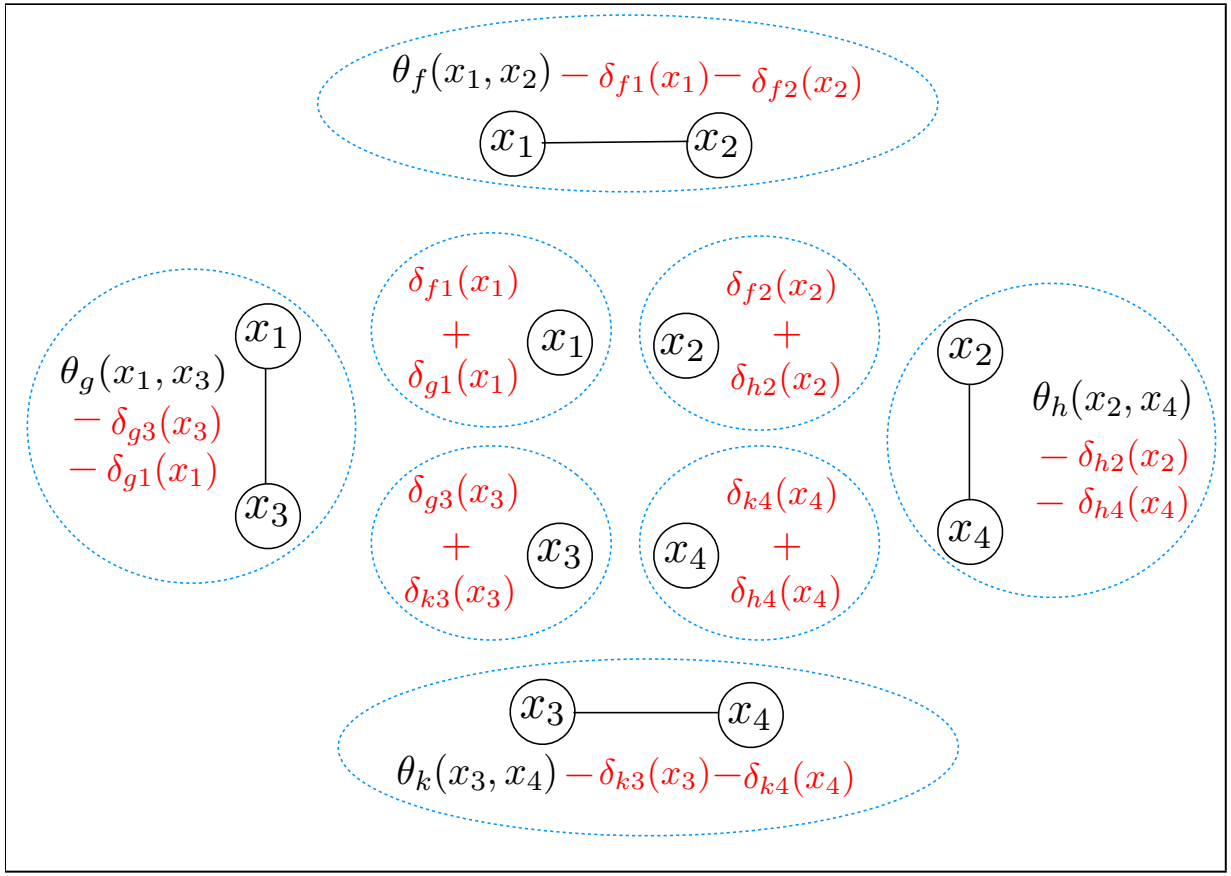
- Note that the right-hand side can be easily evaluated
- One can always *reparameterize* a distribution by operations like

$$\begin{aligned} \theta_i^{\text{new}}(x_i) &= \theta_i^{\text{old}}(x_i) + f(x_i) \\ \theta_{ij}^{\text{new}}(x_i, x_j) &= \theta_{ij}^{\text{old}}(x_i, x_j) - f(x_i) \end{aligned}$$

for **any** function $f(x_i)$, without changing the distribution/energy

Dual decomposition





Dual decomposition

- Define:

$$\begin{aligned}\tilde{\theta}_i(x_i) &= \theta_i(x_i) + \sum_{ij \in E} \delta_{j \to i}(x_i) \\ \tilde{\theta}_{ij}(x_i, x_j) &= \theta_{ij}(x_i, x_j) - \delta_{j \to i}(x_i) - \delta_{i \to j}(x_j)\end{aligned}$$

- It is easy to verify that

$$\sum_i \theta_i(x_i) + \sum_{ij \in E} \theta_{ij}(x_i, x_j) = \sum_i \tilde{\theta}_i(x_i) + \sum_{ij \in E} \tilde{\theta}_{ij}(x_i, x_j) \quad \forall \mathbf{x}$$

- Thus, we have that:

$$\text{MAP}(\theta) = \text{MAP}(\tilde{\theta}) \leq \sum_{i \in V} \max_{x_i} \tilde{\theta}_i(x_i) + \sum_{ij \in E} \max_{x_i, x_j} \tilde{\theta}_{ij}(x_i, x_j)$$

- Every value of δ gives a different upper bound on the value of the MAP!
- The **tightest** upper bound can be obtained by minimizing the r.h.s. with respect to δ!

Dual decomposition

- We obtain the following **dual** objective: $L(\delta) =$

$$\sum_{i \in V} \max_{x_i} \left(\theta_i(x_i) + \sum_{ij \in E} \delta_{j \to i}(x_i) \right) + \sum_{ij \in E} \max_{x_i, x_j} \left(\theta_{ij}(x_i, x_j) - \delta_{j \to i}(x_i) - \delta_{i \to j}(x_j) \right),$$

$$\text{DUAL-LP}(\theta) = \min_{\delta} L(\delta)$$

- This provides an upper bound on the MAP assignment!

$$\text{MAP}(\theta) \quad \leq \quad \text{DUAL-LP}(\theta) \leq L(\delta)$$

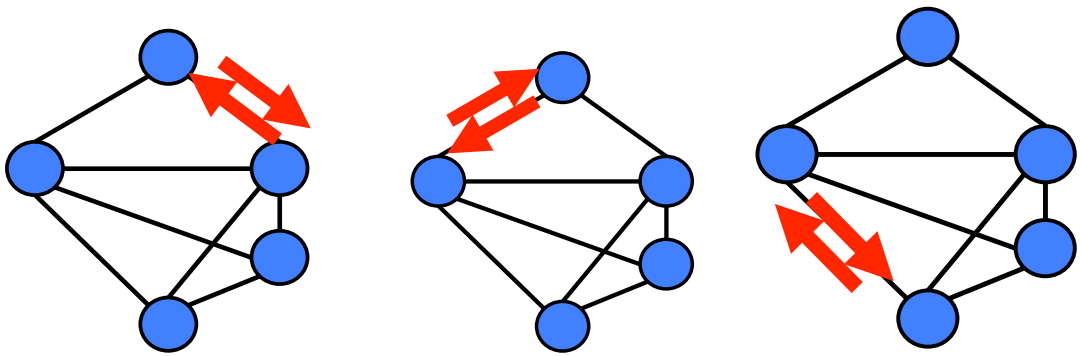
- How can find δ which give tight bounds?

Solving the dual efficiently

- Many ways to solve the dual linear program, i.e. minimize with respect to δ:

$$\sum_{i \in V} \max_{x_i} \left(\theta_i(x_i) + \sum_{ij \in E} \delta_{j \to i}(x_i) \right) + \sum_{ij \in E} \max_{x_i, x_j} \left(\theta_{ij}(x_i, x_j) - \delta_{j \to i}(x_i) - \delta_{i \to j}(x_j) \right),$$

- One option is to use the subgradient method
- Can also solve using **block coordinate-descent**, which gives algorithms that look very much like max-sum belief propagation:



Max-product linear programming (MPLP) algorithm

Input: A set of factors $\theta_i(x_i), \theta_{ij}(x_i, x_j)$

Output: An assignment $x_1, \ldots, x_n$ that approximates the MAP

Algorithm:

- Initialize $\delta_{i \to j}(x_j) = 0, \quad \delta_{j \to i}(x_i) = 0, \quad \forall ij \in E, x_i, x_j$
- Iterate until small enough change in $L(\delta)$:

  For each edge $ij \in E$ (sequentially), perform the updates:

$$\delta_{j \to i}(x_i) = -\frac{1}{2}\delta_i^{-j}(x_i) + \frac{1}{2}\max_{x_j}\left[\theta_{ij}(x_i, x_j) + \delta_j^{-i}(x_j)\right] \quad \forall x_i$$

$$\delta_{i \to j}(x_j) = -\frac{1}{2}\delta_j^{-i}(x_j) + \frac{1}{2}\max_{x_i}\left[\theta_{ij}(x_i, x_j) + \delta_i^{-j}(x_i)\right] \quad \forall x_j$$

  where $\delta_i^{-j}(x_i) = \theta_i(x_i) + \sum_{ik \in E, k \neq j} \delta_{k \to i}(x_i)$

- Return $x_i \in \arg\max_{\hat{x}_i} \tilde{\theta}_i^{\delta}(\hat{x}_i)$

Generalization to arbitrary factor graphs

Inputs:

- A set of factors $\theta_i(x_i), \theta_f(\boldsymbol{x}_f)$.

Output:

- An assignment $x_1, \ldots, x_n$ that approximates the MAP.

Algorithm:

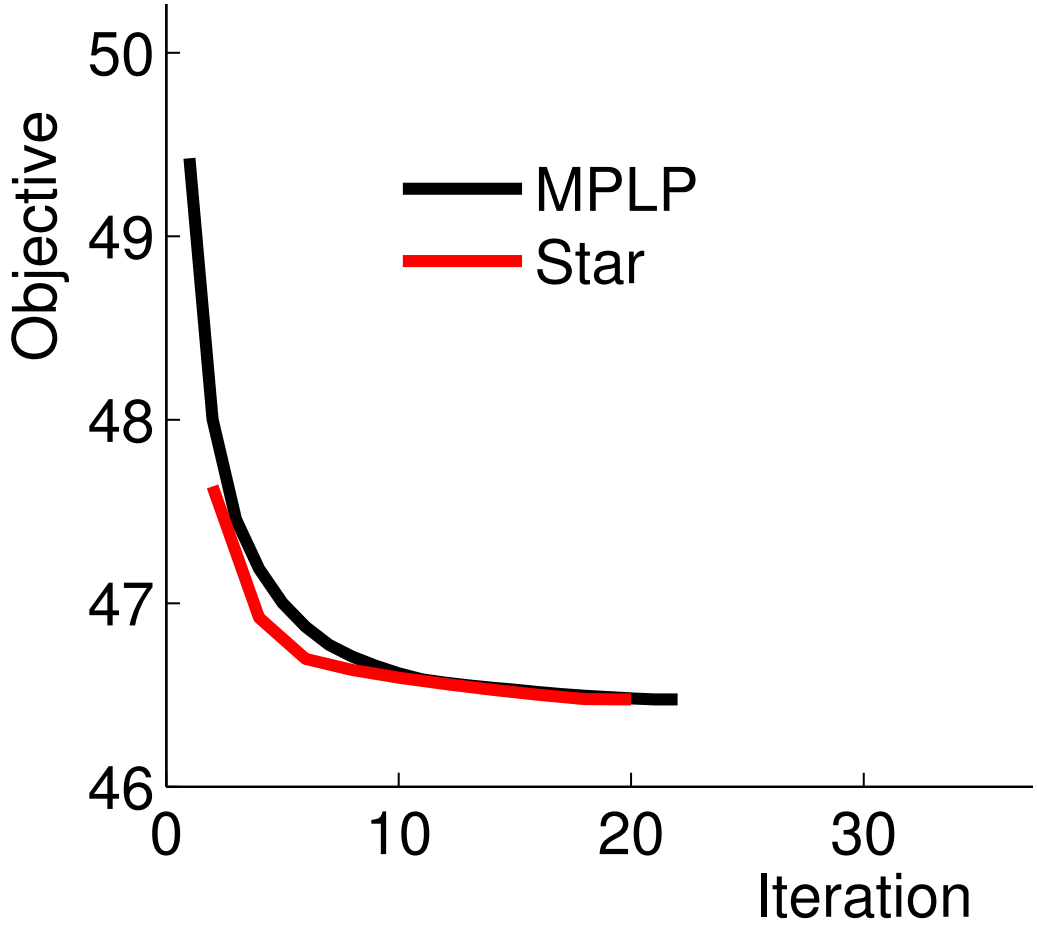
- Initialize $\delta_{fi}(x_i) = 0, \quad \forall f \in F, i \in f, x_i$.
- Iterate until small enough change in $L(\boldsymbol{\delta})$ (see Eq. 1.2):
  For each $f \in F$, perform the updates

$$\delta_{fi}(x_i) = -\delta_i^{-f}(x_i) + \frac{1}{|f|} \max_{\boldsymbol{x}_{f \setminus i}} \left[\theta_f(\boldsymbol{x}_f) + \sum_{\hat{i} \in f} \delta_{\hat{i}}^{-f}(x_{\hat{i}}) \right], \tag{1.16}$$

  simultaneously for all $i \in f$ and x_i. We define $\delta_i^{-f}(x_i) = \theta_i(x_i) + \sum_{\hat{f} \neq f} \delta_{\hat{f}i}(x_i)$.
- Return $x_i \in \arg\max_{\hat{x}_i} \bar{\theta}_i^{\boldsymbol{\delta}}(\hat{x}_i)$ (see Eq. 1.6).

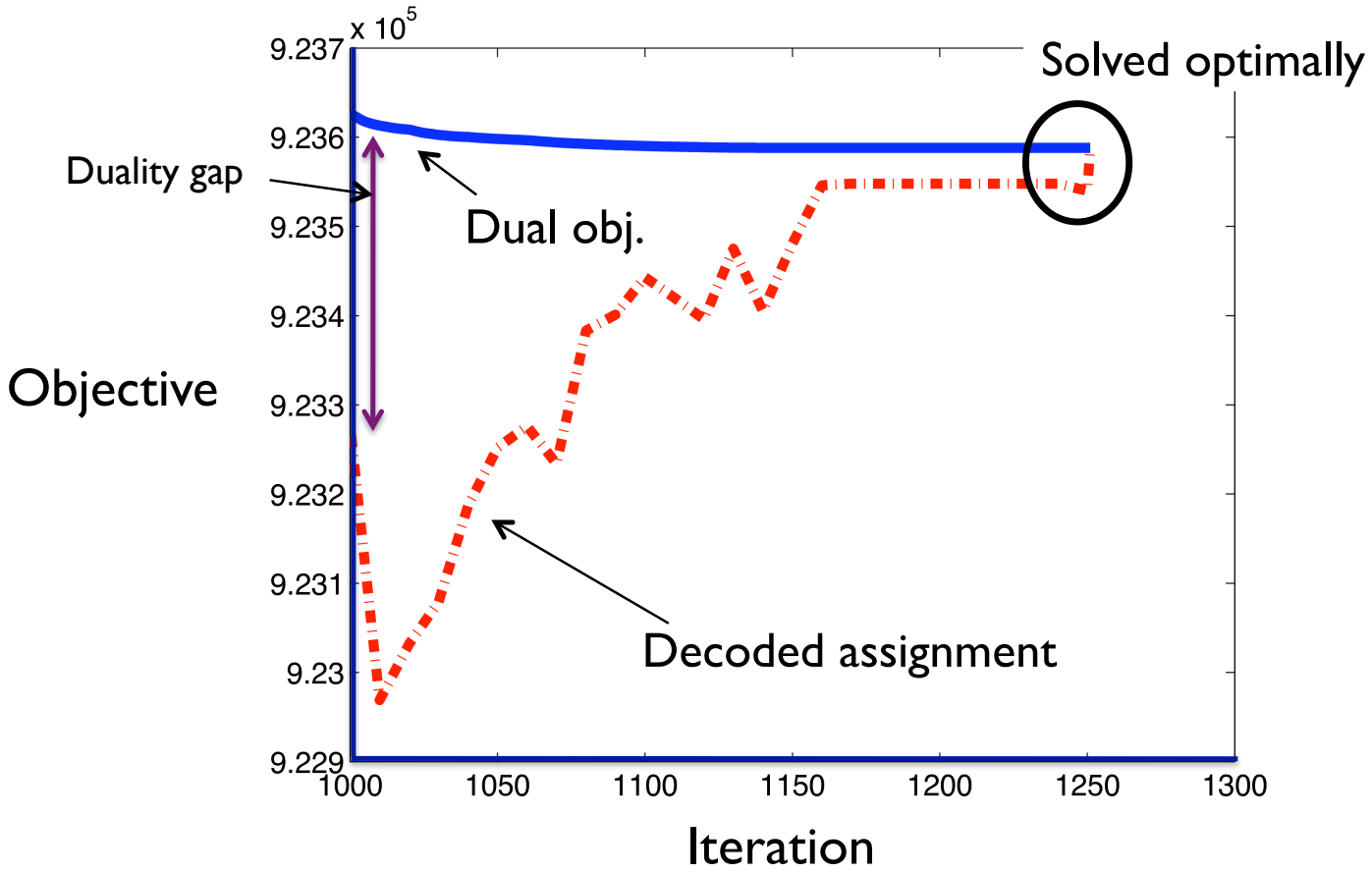
Experimental results

Comparison of two block coordinate descent algorithms on a 10×10 node Ising grid:



Experimental results

Performance on stereo vision inference task:



Today's lecture

1. Dual decomposition
2. MAP inference as an integer linear program
3. Linear programming relaxations for MAP inference

Dual decomposition = LP relaxation

- Recall we obtained the following **dual** linear program: $L(\delta) =$

$$\sum_{i \in V} \max_{x_i} \left(\theta_i(x_i) + \sum_{ij \in E} \delta_{j \to i}(x_i) \right) + \sum_{ij \in E} \max_{x_i, x_j} \left(\theta_{ij}(x_i, x_j) - \delta_{j \to i}(x_i) - \delta_{i \to j}(x_j) \right),$$

$$\text{DUAL-LP}(\theta) = \min_{\delta} L(\delta)$$

- We showed two ways of upper bounding the value of the MAP assignment:

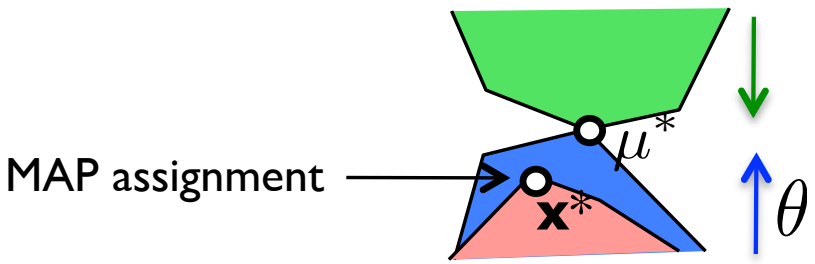
$$\begin{aligned} \text{MAP}(\theta) &\leq \text{LP}(\theta) && (1) \\ \text{MAP}(\theta) &\leq \text{DUAL-LP}(\theta) \leq L(\delta) && (2) \end{aligned}$$

- Although we derived these linear programs in seemingly very different ways, in turns out that:

$$\text{LP}(\theta) = \text{DUAL-LP}(\theta)$$

- The dual LP allows us to upper bound the value of the MAP assignment without solving a LP to optimality

Linear programming duality



(Dual) LP relaxation

(Primal) LP relaxation

Integer linear program

$$\text{MAP}(\theta) \leq \text{LP}(\theta) = \text{DUAL-LP}(\theta) \leq L(\delta)$$

How to solve integer linear programs?

- Local search
  - Start from an arbitrary assignment (e.g., random). Iterate:
  - Choose a variable. Change a new state for this variable to maximize the value of the resulting assignment
- Branch-and-bound
  - Exhaustive search over space of assignments, pruning branches that can be provably shown not to contain a MAP assignment
  - Can use the LP relaxation or its dual to obtain upper bounds
  - Lower bound obtained from value of any assignment found
- Branch-and-cut (most powerful method; used by CPLEX & Gurobi)
  - Same as branch-and-bound; spend more time getting tighter bounds
  - Adds *cutting-planes* to cut off fractional solutions of the LP relaxation, making the upper bound tighter