Inference and Representation

David Sontag

New York University

Lecture 14, Dec. 15, 2015

Final exam

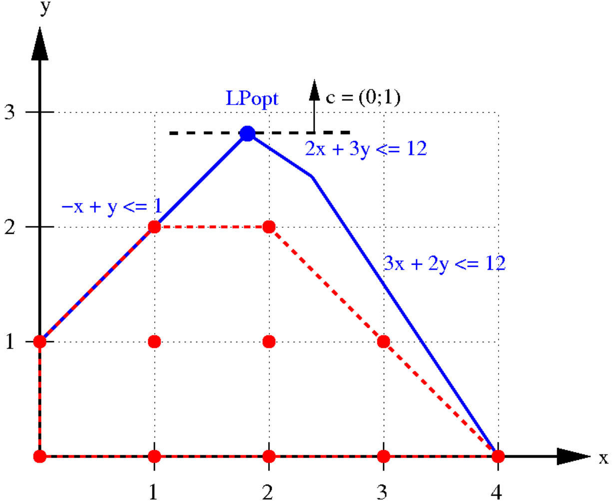
- I will hold office hours this Thursday, 3:30pm. Bring your exam-related questions!
- Final exam in class next week. Closed book; no calculators/phones/computers
- Final covers everything up to and including this week's lab (12/16)

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

Integer linear programming

$\max y$ subject to: $-x + y \leq 1$; $3x + 2y \leq 12$; $2x + 3y \leq 12$; $x, y \in \mathbb{Z}_+$



(Source: Wikipedia)

Integer linear programming

Applications:

- Production planning
- Scheduling (e.g., assigning buses or subways to routes)
- Telecommunication networks
- Bayesian network structure learning

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)

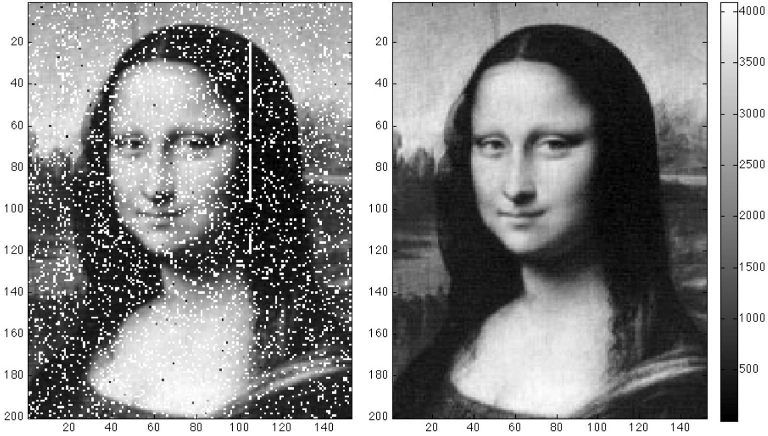
- Furthermore, since log is monotonic, letting $\theta_c(\mathbf{x_c}) = \lg\phi_c(\mathbf{x_c})$, we have that this is equivalent to

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

  (called *max-sum*)

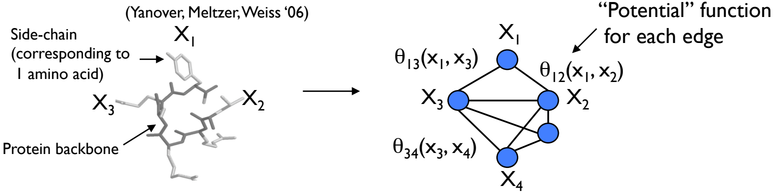
Motivating application: image denoising

- Input (left): noisy image
- Output (right): denoised image



Motivating application: protein side-chain placement

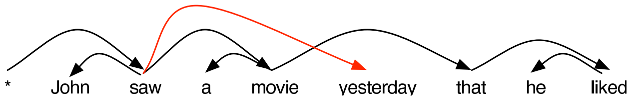
- Find "minimum energy" conformation of amino acid side-chains along a fixed carbon backbone:



- 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

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

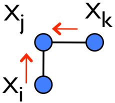
- What is the dimension of μ, if binary variables?

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)

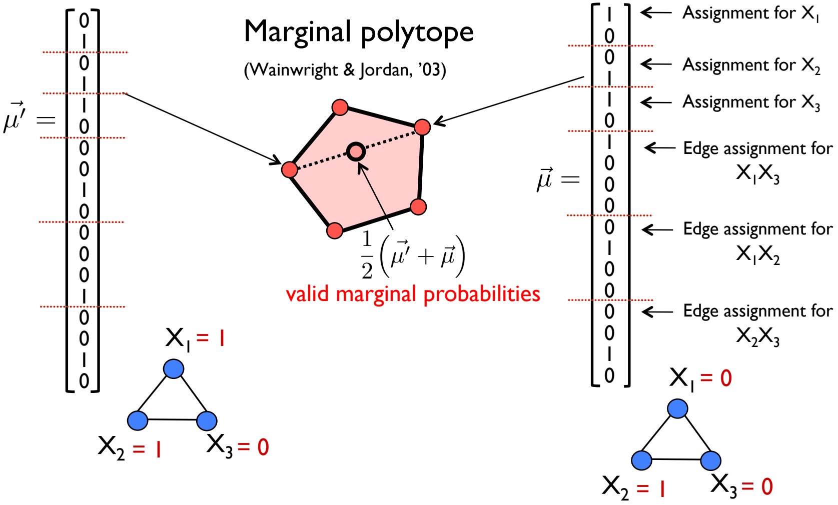
$$\mathrm{MAP}(\theta) = \max_{\mu} \sum_{i \in V} \sum_{x_i} \theta_i(x_i)\mu_i(x_i) + \sum_{ij \in E} \sum_{x_i, x_j} \theta_{ij}(x_i, x_j)\mu_{ij}(x_i, x_j)$$

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

Visualization of integer μ vectors



Linear programming relaxation for MAP

Integer linear program was:

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

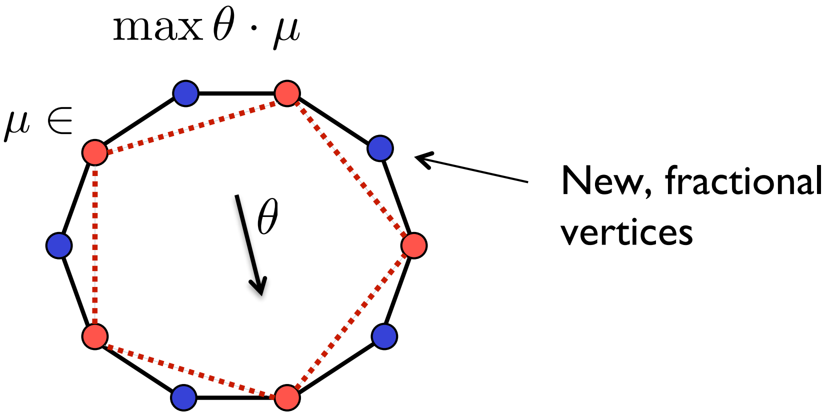
subject to

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

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

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

LP relaxation optimizes over larger feasible space



Linear programming relaxation for MAP

Linear programming relaxation is:

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

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

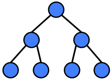
- Linear programs can be solved **efficiently**! Simplex method, interior point, ellipsoid algorithm
- Since the LP relaxation maximizes over a **larger** set of solutions, its value can only be *higher*

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

- LP relaxation is **tight** for tree-structured MRFs. Related to PS5, Q1.

Local consistency constraints are *exact* for trees

- **Theorem:** The local consistency constraints *exactly* define the marginal polytope for a tree-structured MRF:



- **Proof:** Consider any $\vec{\mu} \in M_L$. We specify a distribution $p_T(\mathbf{x})$ for which $\mu_i(x_i)$ and $\mu_{ij}(x_i, x_j)$ are the pairwise and singleton marginals of the distribution p_T
- Let X_1 be the root of the tree, and direct edges away from root. Then,

$$p_T(\mathbf{x}) = \mu_1(x_1) \prod_{i \in V \setminus X_1} \frac{\mu_{i,pa(i)}(x_i, x_{pa(i)})}{\mu_{pa(i)}(x_{pa(i)})} = \prod_{(i,j) \in T} \frac{\mu_{ij}(x_i, x_j)}{\mu_i(x_i)\mu_j(x_j)} \prod_{j \in V} \mu_j(x_j).$$

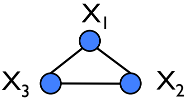
- Because of the local consistency constraints, each term in the product can be interpreted as a conditional probability.

Example for non-tree models

- For non-trees, the local consistency constraints are an *outer bound* on the marginal polytope
- Example of $\vec{\mu} \in M_L \backslash M$ for a MRF on binary variables:

$\mu_{ij}(x_i, x_j) =$

	$X_j = 0$	$X_j = 1$
$X_i = 0$	0	.5
$X_i = 1$	.5	0



- To see that this is not in M, note that it violates the following triangle inequality (valid for marginals of MRFs on **binary variables**):

$$\sum_{x_1 \neq x_2} \mu_{1,2}(x_1, x_2) + \sum_{x_2 \neq x_3} \mu_{2,3}(x_2, x_3) + \sum_{x_1 \neq x_3} \mu_{1,3}(x_1, x_3) \leq 2.$$

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:

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

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

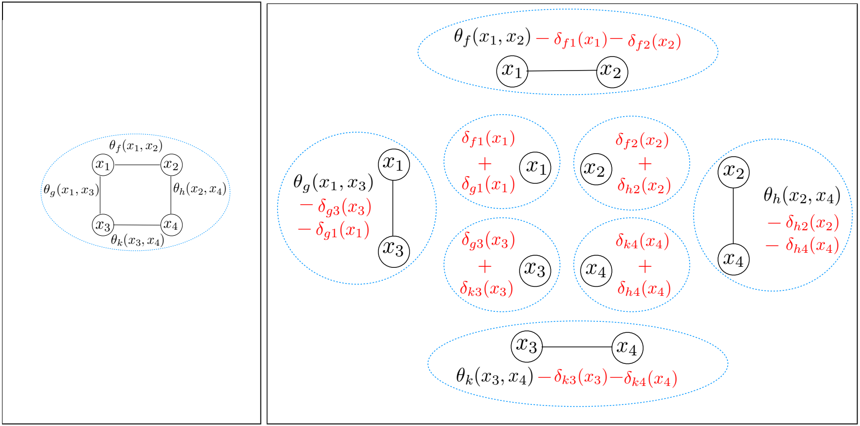
$$\mathrm{MAP}(\theta) \le \sum_{i \in V} \max_{x_i} \theta_i(x_i) + \sum_{ij \in E} \max_{x_i, x_j} \theta_{ij}(x_i, x_j)$$

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

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

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:

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

- 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: $L(\delta) =$

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

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

- This provides an upper bound on the MAP assignment!

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

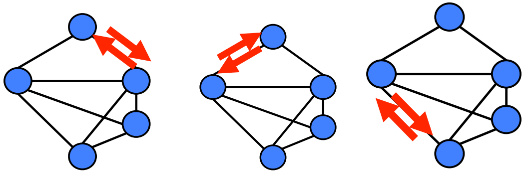
- How can find δ which give tight bounds?

Solving the dual efficiently

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

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

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

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

  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:

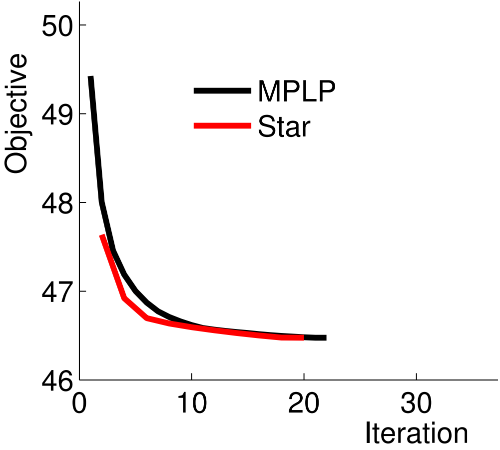
- Initialize $\delta_{fi}(x_i) = 0, \quad \forall f \in F, i \in f, x_i$.
- Iterate until small enough change in $L(\boldsymbol{\delta})$ (see Eq. 1.2):
  For each $f \in F$, perform the updates

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

  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).

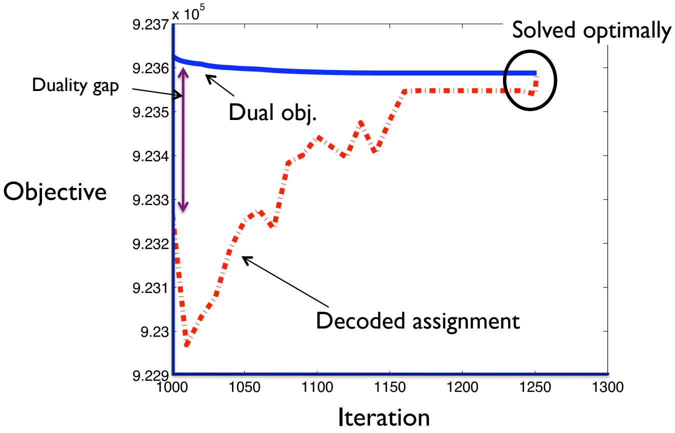
Experimental results

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



Experimental results

Performance on stereo vision inference task:



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:

$$\text{MAP}(\theta) \leq \text{LP}(\theta) \quad (1)$$

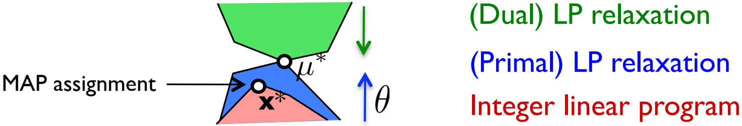
$$\text{MAP}(\theta) \leq \text{DUAL-LP}(\theta) \leq L(\delta) \quad (2)$$

- 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



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

How to solve integer linear programs?

- Local search (iterated conditional modes)
  - 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

Cutting-plane algorithm

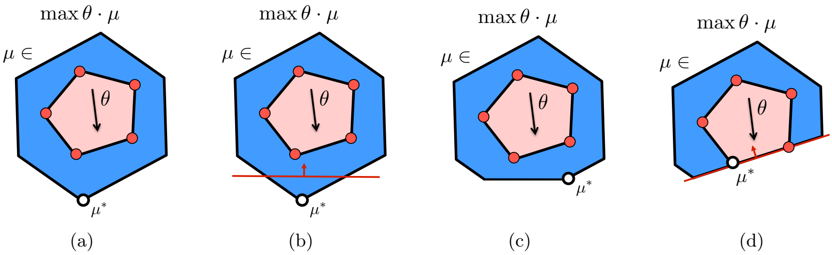


Figure 2-6: Illustration of the cutting-plane algorithm. (a) Solve the LP relaxation. (b) Find a violated constraint, add it to the relaxation, and repeat. (c) Result of solving the tighter LP relaxation. (d) Finally, we find the MAP assignment.

Course evaluation

That's it, folks! Thanks for a great semester. Please stay and fill out the course evaluation.