An Introduction to LP Relaxations for MAP Inference

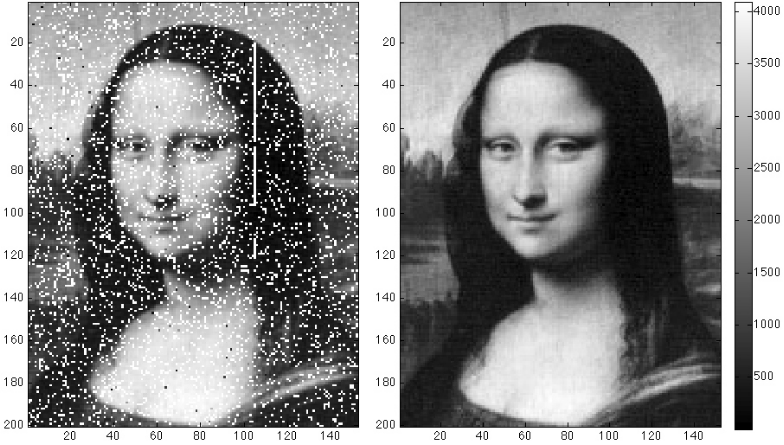
Adrian Weller

Machine Learning Reading Group
Oct 22, 2015

With thanks to David Sontag (NYU)
for use of some of his slides and illustrations

Example of MAP inference: image denoising

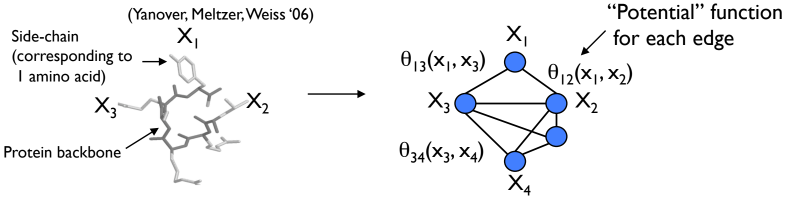
Inference is combining prior beliefs with observed evidence to form a prediction.



$\longrightarrow$ MAP inference

Example of MAP inference: protein side-chain placement

- Find "minimum energy" configuration 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)

Outline of talk

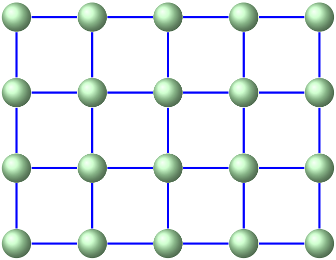
- Background on undirected graphical models
- Basic LP relaxation
- Tighter relaxations
- Message passing and dual decomposition

We'll comment on

- When is an LP relaxation tight
- Relationship to marginal inference

Background: *undirected graphical models*

- Powerful way to represent relationships across variables
- Many applications including: computer vision, social network analysis, deep belief networks, protein folding...
- In this talk, focus on pairwise models with discrete variables (sometimes *binary*)



Example: Grid for computer vision

Background: *undirected graphical models*

- Discrete variables $X_1, \ldots, X_n$ with $X_i \in \{0, \ldots, k_i - 1\}$
- Potential functions, will somehow write as vector θ
- Write $x = (\ldots x_1, \ldots, x_n, \ldots)$ for one complete configuration of all variables, $\theta \cdot x$ for its total score
- Probability distribution given by

$$p(x) = \frac{1}{Z} \exp(\theta \cdot x)$$

- To ensure probabilities sum to 1, need normalizing constant or *partition function* $Z = \sum_x \exp(\theta \cdot x)$
- We are interested in *maximum a posteriori (MAP) inference* i.e., find a global configuration with highest probability

$$x^* \in \arg\max p(x) = \arg\max \theta \cdot x$$

Background: *how do we write potentials as a vector θ?*

- Intuitively, $\theta \cdot x$ means the total score of a configuration x, where we sum over all potential functions
- Abusing notation slightly, if we have potential functions θ_c over some subsets $c \in C$ of variables, then we want $\sum_{c \in C} \theta_c(x_c)$, where x_c means a configuration of variables just in the subset c
- $\theta_c(x_c)$ provides a measure of local compatibility, a table of values

Background: *how do we write potentials as a vector θ?*

- Intuitively, $\theta \cdot x$ means the total score of a configuration x, where we sum over all potential functions
- Abusing notation slightly, if we have potential functions θ_c over some subsets $c \in C$ of variables, then we want $\sum_{c \in C} \theta_c(x_c)$, where x_c means a configuration of variables just in the subset c
- $\theta_c(x_c)$ provides a measure of local compatibility, a table of values
- If we only have some unary/singleton potentials θ_i and edge/pairwise potentials θ_{ij} then we can write the total score as

$$\sum_i \theta_i(x_i) + \sum_{(i,j)} \theta_{(i,j)}(x_i, x_j)$$

- Indices? Usually assume either no unary potentials (absorb them into edges) or one for every variable, leading to a graph topology (V, E) with total score

$$\sum_{i \in V = \{1,\dots,n\}} \theta_i(x_i) + \sum_{(i,j) \in E} \theta_{(i,j)}(x_i, x_j)$$

Background: *overcomplete representation*

The overcomplete representation conveniently allows us to write

$$\theta \cdot x = \sum_{i \in V} \theta_i(x_i) + \sum_{(i,j) \in E} \theta_{(i,j)}(x_i, x_j)$$

- Have a sufficient statistic for each possible configuration of every potential
  $x = (\mathbb{1}[X_1 = 0], \mathbb{1}[X_1 = 1], \ldots, \mathbb{1}[X_n = 0], \mathbb{1}[X_n = 1], \ldots,$
  $\mathbb{1}[X_i = 0, X_j = 0], \mathbb{1}[X_i = 0, X_j = 1], \mathbb{1}[X_i = 1, X_j = 0], \mathbb{1}[X_i = 1, X_j = 1], \ldots)$
  $\theta = (\theta_1(0), \theta_1(1), \ldots, \theta_n(0), \theta_n(1), \ldots,$
  $\theta_{ij}(0,0), \theta_{ij}(0,1), \theta_{ij}(1,0), \theta_{ij}(1,1), \ldots)$
- There are many possible values of x *how many?*

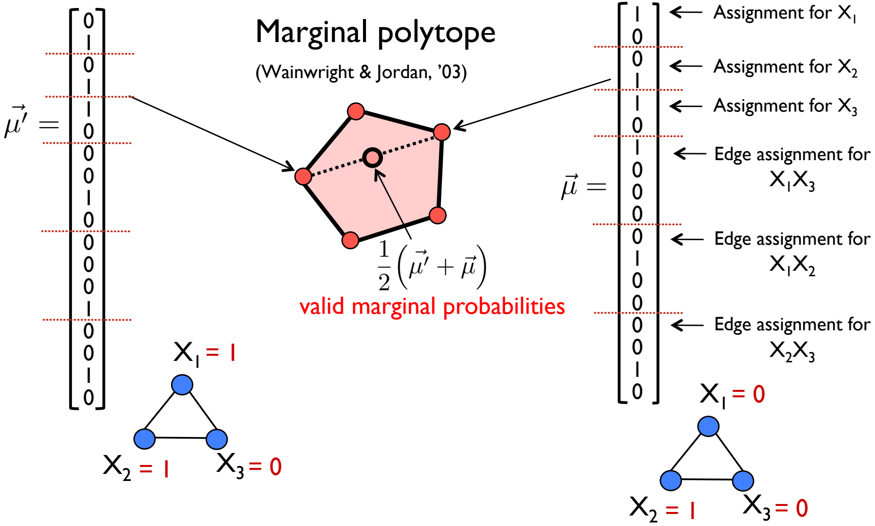
Background: *overcomplete representation*

The overcomplete representation conveniently allows us to write

$$\theta \cdot x = \sum_{i \in V} \theta_i(x_i) + \sum_{(i,j) \in E} \theta_{(i,j)}(x_i, x_j)$$

- Have a sufficient statistic for each possible configuration of every potential
  $x = (\mathbb{1}[X_1 = 0], \mathbb{1}[X_1 = 1], \ldots, \mathbb{1}[X_n = 0], \mathbb{1}[X_n = 1], \ldots,$
  $\mathbb{1}[X_i = 0, X_j = 0], \mathbb{1}[X_i = 0, X_j = 1], \mathbb{1}[X_i = 1, X_j = 0], \mathbb{1}[X_i = 1, X_j = 1], \ldots)$
  $\theta = (\theta_1(0), \theta_1(1), \ldots, \theta_n(0), \theta_n(1), \ldots,$
  $\theta_{ij}(0,0), \theta_{ij}(0,1), \theta_{ij}(1,0), \theta_{ij}(1,1), \ldots)$
- There are many possible values of x *how many?* $\prod_{i \in V} k_i$
- Can consider a distribution over them

Background: *the marginal polytope* (all valid marginals)



Background: *overcomplete and minimal representations*

- The overcomplete representation is highly redundant,
  e.g. $\mu_i(0) + \mu_i(1) = 1 \; \forall i$
- How many dimensions if n binary variables with m edges?

Background: *overcomplete and minimal representations*

- The overcomplete representation is highly redundant, e.g. $\mu_i(0) + \mu_i(1) = 1 \; \forall i$
- How many dimensions if n binary variables with m edges? $2n + 4m$
- Instead, we sometimes pick a minimal representation
- What's the minimum number of dimensions we need?

Background: *overcomplete and minimal representations*

- The overcomplete representation is highly redundant,
  e.g. $\mu_i(0) + \mu_i(1) = 1 \ \forall i$
- How many dimensions if n binary variables with m edges? $2n + 4m$
- Instead, we sometimes pick a minimal representation
- What's the minimum number of dimensions we need? $n + m$
- For example, we could use $q = (q_1, \ldots, q_n, \ldots, q_{ij}, \ldots)$ where $q_i = \mu_i(1) \ \forall i, q_{ij} = \mu_{ij}(1,1) \ \forall (i,j)$, then

$$\mu_i = \begin{pmatrix} 1 - q_i \\ q_i \end{pmatrix}, \mu_j = \begin{pmatrix} 1 - q_j \\ q_j \end{pmatrix}, \quad \mu_{ij} = \begin{pmatrix} 1 + q_{ij} - q_i - q_j & q_j - q_{ij} \\ q_i - q_{ij} & q_{ij} \end{pmatrix}$$

- Note many other possible minimal representations

LP relaxation: MAP as an *integer linear program* (ILP)

- MAP inference as a discrete optimization problem is to identify a configuration with maximum total score

$$
\begin{aligned}
x^* &\in \arg\max_x \sum_{i \in V} \theta_i(x_i) + \sum_{ij \in E} \theta_{ij}(x_i, x_j) \\
&= \arg\max_x \theta \cdot x \\
&= \arg\max_\mu \sum_{i \in V} \sum_{x_i} \theta_i(x_i)\mu_i(x_i) + \sum_{ij \in E} \sum_{x_i, x_j} \theta_{ij}(x_i, x_j)\mu_{ij}(x_i, x_j) \\
&= \arg\max_\mu \theta \cdot \mu \qquad \text{s.t. } \mu \text{ is integral.}
\end{aligned}
$$

- Other constraints?

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 assignments are 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)

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

subject to:

$$\begin{array}{rcll} \mu_i(x_i) & \in & \{0,1\} & \forall i \in V, x_i \quad \text{(edge terms?)} \\ \sum_{x_i} \mu_i(x_i) & = & 1 & \forall i \in V \\ \mu_i(x_i) & = & \sum_{x_j} \mu_{ij}(x_i, x_j) & \forall ij \in E, x_i \\ \mu_j(x_j) & = & \sum_{x_i} \mu_{ij}(x_i, x_j) & \forall ij \in E, x_j \end{array}$$

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

Linear programming (LP) relaxation for MAP

Integer linear program was:

$$\mathrm{MAP}(\theta) = \max_{\mu} \theta \cdot \mu$$

subject to

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

Now relax integrality constraints, allow variables to be **between** 0 and 1:

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

Basic LP relaxation for MAP

$$
\begin{aligned}
\mathrm{LP}(\theta) &= \max_{\mu} \theta \cdot \mu \\
\text{s.t.} \quad \mu_i(x_i) &\in [0,1] \quad \forall i \in V, x_i \\
\sum_{x_i} \mu_i(x_i) &= 1 \quad \forall i \in V \\
\mu_i(x_i) &= \sum_{x_j} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_i \\
\mu_j(x_j) &= \sum_{x_i} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_j
\end{aligned}
$$

- Linear programs can be solved efficiently: simplex, interior point, ellipsoid algorithm
- Since the LP relaxation maximizes over a larger set, its value can only be higher

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

The *local polytope*

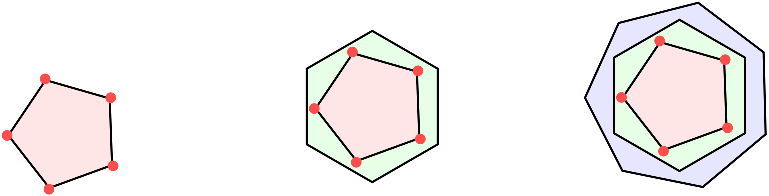
$$
\begin{aligned}
\mathrm{LP}(\theta) &= \max_{\mu} \theta \cdot \mu \\
\text{s.t.} \quad \mu_i(x_i) &\in [0,1] \quad \forall i \in V, x_i \\
\sum_{x_i} \mu_i(x_i) &= 1 \quad \forall i \in V \\
\mu_i(x_i) &= \sum_{x_j} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_i \\
\mu_j(x_j) &= \sum_{x_i} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_j
\end{aligned}
$$

- All these constraints are linear
- Hence define a polytope in the space of marginals
- Here we enforced only local (pairwise) consistency, which defines the *local polytope*
- If instead we had optimized over the marginal polytope, which enforces global consistency, then we would have MAP(θ) =LP(θ), i.e. the LP is tight *why? why don't we do this?*

Tighter relaxations of the marginal polytope

- Enforcing consistency of pairs of variables leads to the local polytope $\mathbb{L}_2$
- The marginal polytope enforces consistency over *all variables* $\mathbb{M} = \mathbb{L}_n$
- Natural to consider the Sherali-Adams hierarchy of successively tighter relaxations $\mathbb{L}_r$ $2 \leq r \leq n$ which enforce consistency over clusters of r variables
- Just up from the local polytope is the triplet polytope TRI$= \mathbb{L}_3$
- Can be shown that for binary variables, TRI$=$CYC, the cycle polytope, which enforces consistency over all cycles
  In general, TRI $\subseteq$ CYC, open problem if TRI $=$ CYC [SonPhD §3]

Stylized illustration of polytopes



marginal polytope
global consistency

cycle polytope
cycle consistency

local polytope
pair consistency

More accurate $\leftrightarrow$ Less accurate

More computationally intensive $\leftrightarrow$ Less computationally intensive

When is the LP tight?

- For a model without cycles, local polytope $\mathbb{L}_2 = \mathbb{M}$ marginal polytope, hence the basic LP ('first order') is always tight
- More generally, if a model has treewidth r then LP+$\mathbb{L}_{r+1}$ is tight [WJ04] TOPOLOGY
- Separately, if we allow any topology but restrict the class of potential functions, interesting results are known POTENTIALS
- For example, the basic LP is tight if all potentials are supermodular
- See fascinating recent work [KolThaZiv15]: if we do not restrict topology, then for any given family of potentials, either the basic LP relaxation is tight or the problem class is NP-hard!
- Identifying hybrid conditions is an exciting current research area

Cutting planes

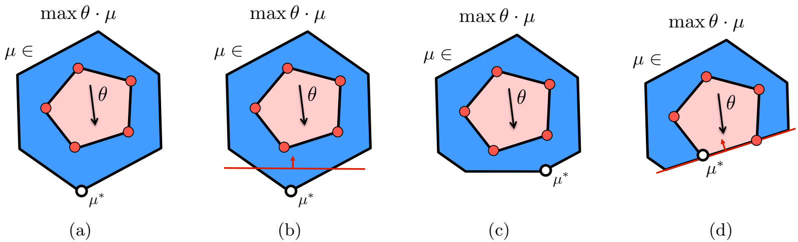
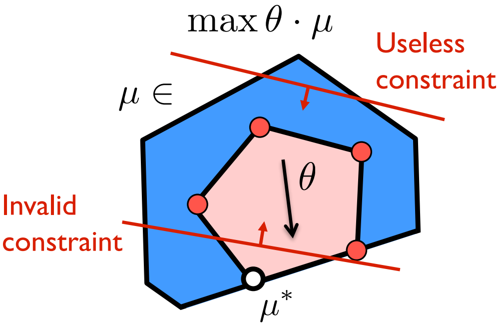


Figure 2-6: Illustration of the cutting-plane algorithm. (a) Solve the LP relaxation. (b) Find a violated constraint, add it to the relaxation, and repeat. (c) Result of solving the tighter LP relaxation. (d) Finally, we find the MAP assignment.

Cutting planes



We want to add constraints that are both *valid* and *useful*

- Valid: does not cut off any integer points
- Useful: leads us to update to a better solution

Methods for solving general integer linear programs

- Local search
  - Start from an arbitrary assignment (e.g., random).
  - Choose a variable.
- 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

Message passing

- Can be a computationally efficient way to obtain or approximate a MAP solution, takes advantage of the graph structure
- Classic example is 'max-product' belief propagation (BP)
- Sufficient conditions are known s.t. this will always converge to the solution of the basic LP, includes that the basic LP is tight [ParkShin-UAI15]
- In general, however, this may not converge to the LP solution (even for supermodular potentials)
- Other methods have been developed, many relate to dual decomposition...

Dual decomposition and reparameterizations

- 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*:

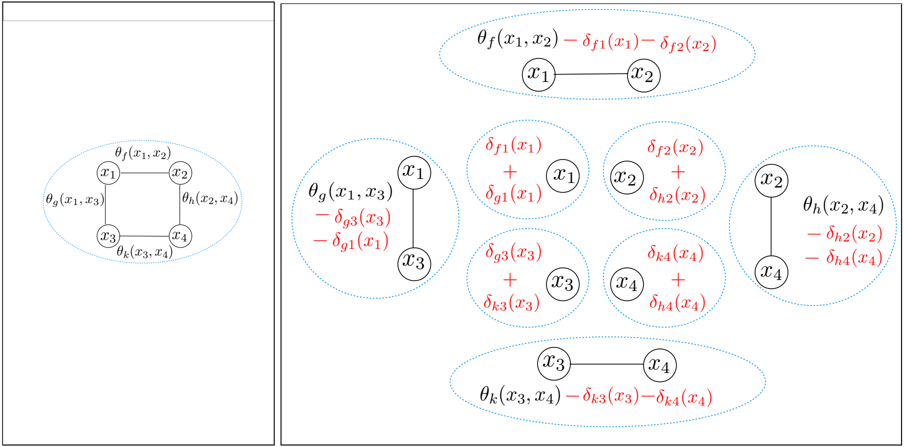
$$\mathrm{MAP}(\theta) \leq \sum_{i \in V} \max_{x_i} \theta_i(x_i) + \sum_{ij \in E} \max_{x_i, x_j} \theta_{ij}(x_i, x_j)$$

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

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

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:

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

- 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 RHS with respect to δ

Dual decomposition

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

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

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

- This provides an upper bound on the MAP assignment

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

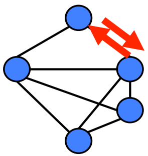
- How can find δ which give tight bounds?

Solving the dual efficiently

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

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

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

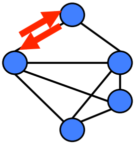


Solving the dual efficiently

- 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 max-sum belief propagation

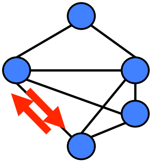


Solving the dual efficiently

- 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 max-sum belief propagation



Max-product linear programming (MPLP) algorithm

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

Output: An assignment $x_1, \dots, x_n$ that approximates a MAP solution

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 [SonGloJaa11]

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:

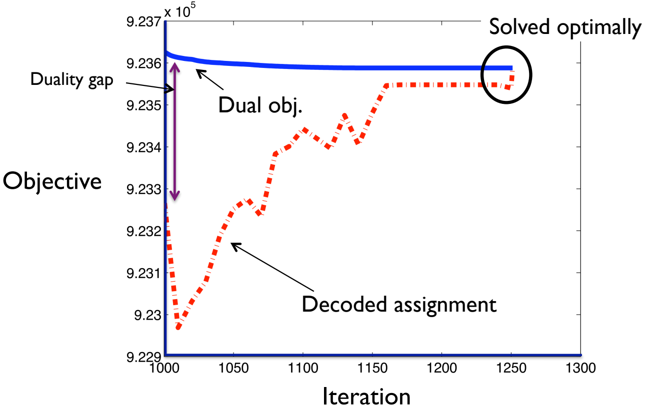
- Initialize $\delta_{fi}(x_i) = 0, \quad \forall f \in F, i \in f, x_i$.
- Iterate until small enough change in $L(\boldsymbol{\delta})$ (see Eq. 1.2):
  For each $f \in F$, perform the updates

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

  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

Performance on stereo vision inference task:



Dual decomposition = basic LP relaxation

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

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

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

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

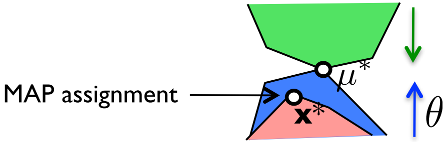
$$\begin{aligned} \text{MAP}(\theta) &\le \text{Basic LP}(\theta) && (1) \\ \text{MAP}(\theta) &\le \text{DUAL-LP}(\theta) \le L(\delta) && (2) \end{aligned}$$

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

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

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

Linear programming duality



(Dual) LP relaxation

(Primal) LP relaxation

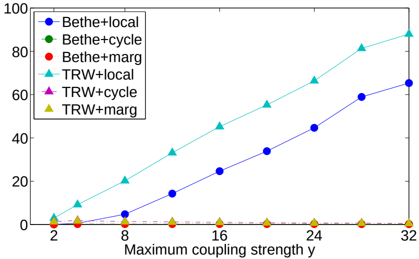
Integer linear program

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

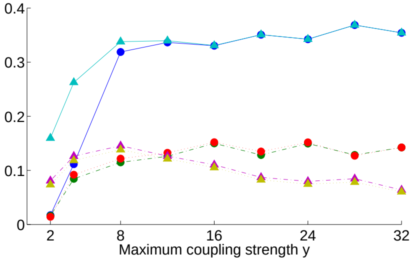
Relationship to marginal inference

- MAP inference: $\mu^* \in \arg\max_{\mu \in \mathbb{M}} \theta \cdot \mu$
- Marginal inference: $\mu^* \in \arg\max_{\mu \in \mathbb{M}} [\theta \cdot \mu + H(\mu)]$
- For marginal inference, both the polytope and the entropy are computationally challenging
- The entropy is often approximated, e.g. by Bethe or TRW entropy
- As before, the marginal polytope $\mathbb{M}$ is typically relaxed to the local polytope $\mathbb{L}$
- Do results for approximate marginal inference improve if we tighten the polytope relaxation?

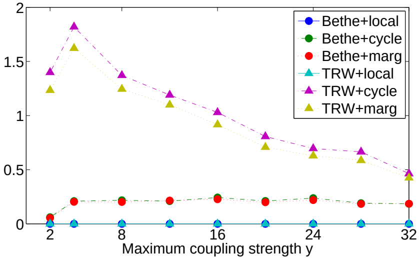
Experiments [WTSJ14]: General models $\theta_i \sim [-2, 2]$ (attractive and repulsive edges) K_{10} topology



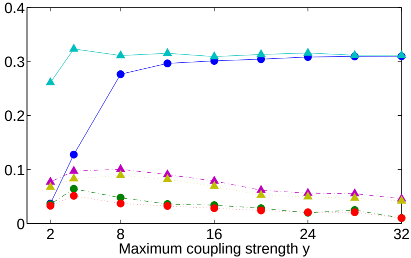
log partition error



Singleton marginals, average ℓ_1 error



log partition error, local zeroed



Pairwise marginals, average ℓ_1 error

Conclusion

- LP relaxations yield a powerful approach for MAP inference
- Naturally lead to
  - considerations of polytope or cutting planes
  - dual decomposition and message passing
- Close relationship to methods for marginal inference
- Help build our understanding as well as develop new algorithmic tools
- Exciting current research

Thank you

References

- V. Kolmogorov, J. Thapper, and S. Živný. The power of linear programming for general-valued CSPs. SIAM Journal on Computing, 44(1):136, 2015.
- S. Park and J. Shin. Max-product belief propagation for linear programming: applications to combinatorial optimization. In *UAI*, 2015.
- D. Sontag. Approximate inference in graphical models using LP relaxations. PhD thesis, MIT, 2010.
- D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. In *Optimization for Machine Learning*, MIT Press, 2011.
- J. Thapper and S. Živný. The complexity of finite-valued CSPs. arxiv technical report, 2015. `http://arxiv.org/abs/1210.2987v3`
- M. Wainwright and M. Jordan. Treewidth-based conditions for exactness of the Sherali-Adams and Lasserre relaxations. Univ. California, Berkeley, Technical Report, 671:4, 2004.
- M. Wainwright and M. Jordan. Graphical models, exponential families and variational inference. Foundations and Trends in Machine Learning, 1(1-2):1305, 2008.
- A. Weller, K. Tang, D. Sontag, and T. Jebara. Understanding the Bethe approximation: When and how can it go wrong? In *UAI*, 2014.