An Introduction to LP Relaxations for MAP Inference

Adrian Weller

MLSALT4 Lecture
Feb 27, 2017

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

For more information, see http://mlg.eng.cam.ac.uk/adrian/
High level overview of our 3 lectures

1. Directed and undirected graphical models (last Friday)
2. LP relaxations for MAP inference (today)
3. Junction tree algorithm for exact inference, belief propagation, variational methods for approximate inference (this Friday)

Further reading / viewing:
- Murphy, *Machine Learning: a Probabilistic Perspective*
- Barber, *Bayesian Reasoning and Machine Learning*
- Bishop, *Pattern Recognition and Machine Learning*
- Koller and Friedman, *Probabilistic Graphical Models*
  [https://www.coursera.org/course/pgm](https://www.coursera.org/course/pgm)
- Wainwright and Jordan, *Graphical Models, Exponential Families, and Variational Inference*
Example of MAP inference: image denoising

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

→ 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
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 $\theta$
- Write $x = (\ldots x_1, \ldots, x_n, \ldots)$ for one ‘overcomplete 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 $\theta$?

- $\theta \cdot x$ means the total score of a configuration $x$, where we sum over all potential functions.
- If we have potential functions $\theta_c$ 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 $\theta$?

- $\theta \cdot x$ means the total score of a configuration $x$, where we sum over all potential functions.
- If we have potential functions $\theta_c$ 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 $\theta_i$ and edge/pairwise potentials $\theta_{ij}$ then we can write the total score as
  \[ \sum_i \theta_i(x_i) + \sum_{(i,j)} \theta_{ij}(x_i, x_j) \]

- 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
  \[ \sum_{i \in V = \{1, \ldots, n\}} \theta_i(x_i) + \sum_{(i,j) \in E} \theta_{ij}(x_i, x_j) \]
Background: \textit{overcomplete representation}

The \textit{overcomplete representation} conveniently allows us to write

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

- Concatenate singleton and edge terms into big vectors

$$\begin{pmatrix} \ldots \\ \ldots \\ \theta_i(0) \\ \theta_i(1) \\ \ldots \\ \theta_{ij}(0,0) \\ \theta_{ij}(0,1) \\ \theta_{ij}(1,0) \\ \theta_{ij}(1,1) \\ \ldots \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \ldots \\ 1[X_i = 0] \\ 1[X_i = 1] \\ \ldots \\ 1[X_i = 0, X_j = 0] \\ 1[X_i = 0, X_j = 1] \\ 1[X_i = 1, X_j = 0] \\ 1[X_i = 1, X_j = 1] \\ \ldots \end{pmatrix}$$

- There are many possible values of $x$ \textit{how many?}
The overcomplete representation conveniently allows us to write

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

- Concatenate singleton and edge terms into big vectors

$$\theta = \begin{pmatrix} \theta_i(0) \\ \theta_i(1) \\ \theta_{ij}(0,0) \\ \theta_{ij}(0,1) \\ \theta_{ij}(1,0) \\ \theta_{ij}(1,1) \\ \cdots \end{pmatrix} \quad x = \begin{pmatrix} \mathbb{1}[X_i = 0] \\ \mathbb{1}[X_i = 1] \\ \mathbb{1}[X_i = 0, X_j = 0] \\ \mathbb{1}[X_i = 0, X_j = 1] \\ \mathbb{1}[X_i = 1, X_j = 0] \\ \mathbb{1}[X_i = 1, X_j = 1] \\ \cdots \end{pmatrix}$$

- There are many possible values of $x$

$$\prod_{i \in V} k_i$$
Background: *Binary pairwise models*

- $\theta \cdot x$ is the score of a configuration $x$
- Probability distribution given by
  \[ p(x) = \frac{1}{Z} \exp(\theta \cdot x) \]

- For MAP inference, want $x^* \in \arg \max p(x) = \arg \max \theta \cdot x$
- Want to optimize over \{0, 1\} coordinates of ‘overcomplete configuration space’ corresponding to all $2^n$ possible settings
- The convex hull of these defines the marginal polytope $\mathcal{M}$
- Each point $\mu \in \mathcal{M}$ corresponds to a probability distribution over the $2^n$ configurations, giving a vector of marginals
Background: *the marginal polytope* (all valid marginals)

![Diagram of the marginal polytope](image)

**Marginal polytope**

(Wainwright & Jordan, '03)

$$\mathbf{\mu} = \frac{1}{2} (\mathbf{\mu}' + \mathbf{\bar{\mu}})$$

valid marginal probabilities

![Assignment for $X_1$](image)

![Assignment for $X_2$](image)

![Assignment for $X_3$](image)

![Edge assignment for $X_1X_2$](image)

![Edge assignment for $X_1X_3$](image)

![Edge assignment for $X_2X_3$](image)

$$X_1 = 1$$

$$X_2 = 1$$

$$X_3 = 0$$

$$X_1 = 0$$

$$X_2 = 1$$

$$X_3 = 0$$
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?
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?
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
MAP inference as a discrete optimization problem is to identify a configuration with maximum total score:

\[
x^* \in \arg \max_x \sum_{i \in V} \theta_i(x_i) + \sum_{ij \in E} \theta_{ij}(x_i, x_j)
\]

\[
= \arg \max_x \theta \cdot x
\]

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

\[
= \arg \max_\mu \theta \cdot \mu \quad \text{s.t. } \mu \text{ is integral}
\]

Any other constraints?
What are the constraints?

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

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

\[
\sum_{x_i} \mu_i(x_i) = 1 \quad \forall i \in V
\]

\[
\mu_{ij}(x_i, x_j) \in \{0, 1\} \quad \forall ij \in E, x_i, x_j
\]

\[
\sum_{x_i, x_j} \mu_{ij}(x_i, x_j) = 1 \quad \forall ij \in E
\]

- Enforce that these assignments are consistent:

\[
\mu_i(x_i) = \sum_{x_j} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_i
\]

\[
\mu_j(x_j) = \sum_{x_i} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_j
\]
MAP as an integer linear program (ILP)

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

$$= \max_{\mu} \theta \cdot \mu$$

subject to:

$$\mu_i(x_i) \in \{0, 1\} \quad \forall i \in V, x_i \quad \text{(edge terms?)}$$

$$\sum_{x_i} \mu_i(x_i) = 1 \quad \forall i \in V$$

$$\mu_i(x_i) = \sum_{x_j} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_i$$

$$\mu_j(x_j) = \sum_{x_i} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_j$$

- Many good off-the-shelf solvers, such as CPLEX and Gurobi
Integer linear program was:

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

subject to

$$\mu_i(x_i) \in \{0, 1\} \quad \forall i \in V, x_i$$

$$\sum_{x_i} \mu_i(x_i) = 1 \quad \forall i \in V$$

$$\mu_i(x_i) = \sum_{x_j} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_i$$

$$\mu_j(x_j) = \sum_{x_i} \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_j$$

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

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

- 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

\[\text{MAP}(\theta) \leq \text{LP}(\theta)\]
The *local polytope*

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

- 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 \(\text{MAP}(\theta) = \text{LP}(\theta)\), i.e. the LP is tight \(\text{why? why don’t we do this?}\)
Tighter relaxations of the marginal polytope

- Enforcing consistency of pairs of variables leads to the local polytope $L_2$
- The marginal polytope enforces consistency over all variables $M = L_n$
- Natural to consider the Sherali-Adams hierarchy of successively tighter relaxations $L_r$ $2 \leq r \leq n$ which enforce consistency over clusters of $r$ variables
- Just up from the local polytope is the triplet polytope $\text{TRI} = L_3$
Stylized illustration of polytopes

marginal polytope $\mathcal{M} = \mathbb{L}_n$

global consistency

triplet polytope $\mathbb{L}_3$

triplet consistency

local polytope $\mathbb{L}_2$

pair consistency

More accurate $\leftrightarrow$ Less accurate

More computationally intensive $\leftrightarrow$ Less computationally intensive
Stylized illustration of polytopes

marginal polytope $\mathbb{M} = \mathbb{L}_n$

global consistency

triplet polytope $\mathbb{L}_3$

triplet consistency

local polytope $\mathbb{L}_2$

pair consistency

More accurate $\leftrightarrow$ Less accurate

More computationally intensive $\leftrightarrow$ Less computationally intensive

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]
When is the LP tight?

- For a model without cycles, the 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 $\text{LP} + \mathbb{L}_{r+1}$ is tight [WaiJor04].

- Separately, if we allow any structure but restrict the class of potential functions, interesting results are known.

- For example, the basic LP is tight if all potentials are supermodular.

- Fascinating recent work [KolThaZiv15]: if we do not restrict structure, 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.
When is MAP inference (relatively) easy?

- Both can be handled efficiently by the basic LP relaxation, $\text{LP} + \mathbb{L}_2$
Cutting planes

\[ \max \theta \cdot \mu \]
\[ \mu \in \theta \]

\( \mu^* \)

(a)

(b)

(c)

(d)

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.
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:

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

- If we push the maximizations \textit{inside} the sums, the value can only \textit{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 \textbf{reparameterize} a distribution by operations like

\[
\begin{align*}
\theta_i^{\text{new}}(x_i) &= \theta_i^{\text{old}}(x_i) + f(x_i) \\
\theta_{ij}^{\text{new}}(x_i, x_j) &= \theta_{ij}^{\text{old}}(x_i, x_j) - f(x_i)
\end{align*}
\]

for \textbf{any} function \( f(x_i) \), without changing the distribution/energy.
Dual decomposition

Introduction to Dual Decomposition for Inference

\[ \theta_f(x_1, x_2) - \delta_{f1}(x_1) - \delta_{f2}(x_2) \]

\[ \theta_g(x_1, x_3) - \delta_{g3}(x_3) - \delta_{g1}(x_1) \]

\[ \theta_k(x_3, x_4) - \delta_{k3}(x_3) - \delta_{k4}(x_4) \]

\[ \theta_h(x_2, x_4) - \delta_{h2}(x_2) - \delta_{h4}(x_4) \]

Figure 1.2: Illustration of the dual decomposition objective.

The original pairwise model consisting of four factors.

The maximization problems corresponding to the objective \( L \).

Each blue ellipse contains the factor to be maximized over. In all figures the singleton terms \( \checkmark_i(x_i) \) are set to zero for simplicity.
Dual decomposition

Define:

\[ \tilde{\theta}_i(x_i) = \theta_i(x_i) + \sum_{ij \in E} \delta_{j \rightarrow i}(x_i) \]

\[ \tilde{\theta}_{ij}(x_i, x_j) = \theta_{ij}(x_i, x_j) - \delta_{j \rightarrow i}(x_i) - \delta_{i \rightarrow j}(x_j) \]

It is easy to verify that

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

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 \( \delta \) 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 \( \delta \).
We obtain the following **dual** objective: 

\[
L(\delta) = \sum_{i \in V} \max_{x_i} \left( \theta_i(x_i) + \sum_{ij \in E} \delta_{j \rightarrow i}(x_i) \right) + \sum_{ij \in E} \max_{x_i, x_j} \left( \theta_{ij}(x_i, x_j) - \delta_{j \rightarrow i}(x_i) - \delta_{i \rightarrow j}(x_j) \right),
\]

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

This provides an upper bound on the MAP assignment

\[
\text{MAP}(\theta) \leq \text{DUAL-LP}(\theta) \leq L(\delta)
\]

How can find \( \delta \) which give tight bounds?
Solving the dual efficiently

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

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

- 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
Solving the dual efficiently

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

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

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

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

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 potentials $\theta_i(x_i), \theta_{ij}(x_i, x_j)$

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

**Algorithm:**

- Initialize $\delta_{i\rightarrow j}(x_j) = 0$, $\delta_{j\rightarrow i}(x_i) = 0$, $\forall ij \in E, x_i, x_j$

- Iterate until small enough change in $L(\delta)$:
  
  For each edge $ij \in E$ (sequentially), perform the updates:

  $\delta_{j\rightarrow i}(x_i) = -\frac{1}{2} \delta_{i\rightarrow j}(x_i) + \frac{1}{2} \max_{x_j}[\theta_{ij}(x_i, x_j) + \delta_{j\rightarrow i}(x_j)]$ \quad $\forall x_i$

  $\delta_{i\rightarrow j}(x_j) = -\frac{1}{2} \delta_{j\rightarrow i}(x_j) + \frac{1}{2} \max_{x_i}[\theta_{ij}(x_i, x_j) + \delta_{i\rightarrow j}(x_i)]$ \quad $\forall x_j$

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

- Return $x_i \in \arg \max_{\hat{x}_i} \tilde{\theta}_i^{\delta}(\hat{x}_i)$
**Inputs:**
- A set of factors $\theta_i(x_i), \theta_f(x_f)$.

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

**Algorithm:**
- Initialize $\delta_{fi}(x_i) = 0, \forall f \in F, i \in f, x_i$.
- Iterate until small enough change in $L(\delta)$ (see Eq. 1.2):
  - For each $f \in F$, perform the updates
    \[
    \delta_{fi}(x_i) = -\delta_i^{-f}(x_i) + \frac{1}{|f|} \max_{x_f \setminus i} \left[ \theta_f(x_f) + \sum_{i \in f} \delta_i^{-f}(x_i) \right],
    \]
    simultaneously for all $i \in f$ and $x_i$. We define $\delta_i^{-f}(x_i) = \theta_i(x_i) + \sum_{f \neq f} \delta_{fi}(x_i)$.
- Return $x_i \in \arg\max_{\hat{x}_i} \tilde{\theta}_i^{\delta}(\hat{x}_i)$ (see Eq. 1.6).
Experimental results

Performance on stereo vision inference task:

- **Objective**
- **Decoded assignment**
- **Dual obj.**
- **Duality gap**

The graph shows the performance over iterations, with the objective and dual objective values plotted against iteration number. The duality gap is indicated, and the point where the objective is solved optimally is marked.

```
Solved optimally
```
Recall we obtained the following dual linear program: 
\[ L(\delta) = \sum_{i \in V} \max_{x_i} \left( \theta_i(x_i) + \sum_{j \in E} \delta_{j \rightarrow i}(x_i) \right) + \sum_{ij \in E} \max_{x_i, x_j} \left( \theta_{ij}(x_i, x_j) - \delta_{j \rightarrow i}(x_i) - \delta_{i \rightarrow j}(x_j) \right), \]

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

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

\[ \text{MAP}(\theta) \leq \text{Basic LP}(\theta) \quad (1) \]
\[ \text{MAP}(\theta) \leq \text{DUAL-LP}(\theta) \leq L(\delta) \quad (2) \]

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

\[ \text{Basic LP}(\theta) = \text{DUAL-LP}(\theta) \quad \text{[SonGloJaa11]} \]

The dual LP allows us to upper bound the value of the MAP assignment without solving an LP to optimality.
Linear programming duality

MAP assignment

(Dual) LP relaxation

(Primal) LP relaxation

Integer linear program

\[ \text{MAP}(\theta) \leq \text{Basic LP}(\theta) = \text{DUAL-LP}(\theta) \leq L(\delta) \]
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.
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 understanding as well as develop new algorithmic tools
- Exciting current research

Thank you
References


