MAP estimation and Linear Programming

Jialin Song

CS 159

04/13/2017

References

This presentation is mainly based on the following papers

- Sections 8.2 and 8.4 of Graphical Models, Exponential Families, and Variational Inference by Wainwright and Jordan [MarJor08]
- Fixing Max-Product: Convergent Message Passing Algorithms for MAP LP-Relaxation [MPLP]

MAP (maximum a posteriori) Inference

Let $G = (V, E)$ be an undirected graph representing an MRF.
For each vertex $v \in V$, a random variable X_v takes value $x_v \in \mathcal{X}_v$.

$p(x|\theta) = \frac{1}{Z} \exp\{\sum_{v \in V} \theta_v(x_v) + \sum_{(u,v) \in E} \theta_{uv}(x_u, x_v)\}$ where Z is the partition function.

Goal: find
$\arg\max_x p(x|\theta) = \arg\max_x \sum_{v \in V} \theta_v(x_v) + \sum_{(u,v) \in E} \theta_{uv}(x_u, x_v)$

MAP Inference

Alternatively, we can write

$$
\begin{aligned}
&\arg\max_x \sum_{v\in V} \theta_v(x_v) + \sum_{(u,v)\in E} \theta_{uv}(x_u, x_v) \\
&= \arg\max_x \sum_{v\in V} \sum_{x\in\mathcal{X}_v} \mathbb{I}(x = x_v)\theta_v(x_v) + \\
&\sum_{(u,v)\in E} \sum_{(x_1,x_2)\in\mathcal{X}_u\times\mathcal{X}_v} \mathbb{I}(x_1 = x_u, x_2 = x_v)\theta_{uv}(x_u, x_v) \\
&:= \arg\max_x \langle\theta, \phi(x)\rangle \\
&= \arg\max_{\mu\in\mathcal{M}(G)} \langle\theta, \mu\rangle \text{ See [WaiJor08, Theorem 8.1]}
\end{aligned}
$$

where $\phi(x)$ is an indicator function on x, and $\mathcal{M}(G)$ is the **marginal polytope**.

Marginal Polytope

$\mathcal{M}(G) := \{\mu | \mu_s(x_s) = \sum_{x_t, t \neq s} p_\mu(\mathbf{x}), \forall s \in V; \mu_{st}(x_s, x_t) = \sum_{x_u, u \neq s,t} p_\mu(\mathbf{x}), \forall (s,t) \in E\}$, for some **probability distribution** p_μ over $\mathbf{x}$.

Important characteristic of $\mathcal{M}$: convex; number of facets exponential in number of variables. As a result, even though

$\arg \max_{\mu \in \mathcal{M}(G)} \langle \theta, \mu \rangle$ is a linear programming problem over a convex set, solving it is still difficult/inefficient.

Relaxation

To make the LP more efficient, we consider relaxing $\mathcal{M}(\mathcal{G})$ to a convex outer bound
$\mathcal{L}(G) := \{\tau \geq 0 \mid \sum_{x_s} \tau_s(x_s) = 1, \forall s \in V; \sum_{x_t} \tau_{st}(x_s, x_t) = \tau_s(x_s), \forall (s,t) \in E\}$

It is easy to verify that for any graph G, $\mathcal{M}(G) \subseteq \mathcal{L}(G)$, but how about the other direction?

Relaxation

As it turns out, if G is a tree, then $\mathcal{M}(G) = \mathcal{L}(G)$. (See Proposition 4.1 from [MarJor8])

However, $\mathcal{M}(G) \neq \mathcal{L}(G)$ for general graphs.

Example [MarJor08]

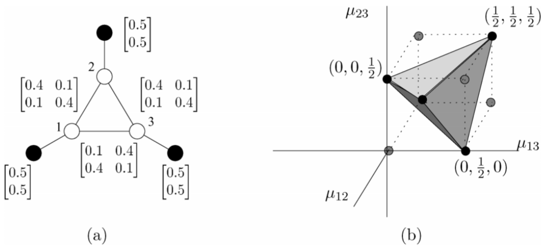

Fig. 4.1 (a) A set of pseudomarginals associated with the nodes and edges of the graph: setting $\beta_{12} = \beta_{23} = 0.4$ and $\beta_{13} = 0.1$ in Equation (4.9) yields a pseudomarginal vector τ which, though locally consistent, is not globally consistent. (b) Marginal polytope $\mathbb{M}(C_3)$ for the three node cycle; in a minimal exponential representation, it is a 6D object. Illustrated here is the slice $\{\mu_1 = \mu_2 = \mu_3 = \frac{1}{2}\}$, as well as the outer bound $\mathbb{L}(C_3)$, also for this particular slice.

Max-product and Linear Programming on Trees

Recall that max-product message passing is another way to do MAP inference, are there connections between max-product and linear programming?

Max-product and Linear Programming on Trees

Recall that max-product message passing is another way to do MAP inference, are there connections between max-product and linear programming?

If G is a tree, yes!

Recall message passing from $t \to s$,
$$M_{ts}(x_s) \leftarrow \max_{x_t \in \mathcal{X}_t} [\exp(\theta_{st}(x_s, x_t) + \theta_t(x_t)) \prod_{u \in N(t) \setminus s} M_{ut}(x_t)].$$
It turns out that for tree-structured graphs, the max-product message passing is a Lagrangian method for solving the dual of the linear program.

LP and its dual

LP in standard form:

$$\begin{aligned} f^* = \sup_{x \in \mathbb{R}^n} & \langle c, x \rangle \\ \text{s.t. } & \langle a_i, x \rangle \geq 0, i = 1, 2, \cdots, k \\ & \langle b_j, x \rangle = 0, j = 1, 2, \cdots, m \end{aligned}$$

The Lagrangian of the LP:
$L(x, \lambda, \mu) := \langle c, x \rangle + \sum_{i=1}^{k} \lambda_i \langle a_i, x \rangle + \sum_{j=1}^{m} \mu_j \langle b_j, x \rangle$, where
$\lambda \in \mathbb{R}_+^k, \mu \in \mathbb{R}^m$.
The dual function:
$h(\lambda, \mu) = \sup_{x \in \mathbb{R}^n} L(x, \lambda, \mu)$
The dual problem:

$$\begin{aligned} d^* = \inf_{\lambda \in \mathbb{R}^k, \mu \in \mathbb{R}^m} & h(\lambda, \mu) \\ \text{s.t. } & \lambda_i \geq 0, i = 1, 2, \cdots, k \end{aligned}$$

LP and its dual

Weak duality: $f^* \leq d^*$
Strong duality: $f^* = d^*$
Strong duality always holds for linear programs.

Max-product and Linear Programming on Trees

Setting up the Lagrangian, for each $x_s \in \mathcal{X}_s$, $\lambda_{st}(x_s)$ is the Lagrangian multiplier associated with the marginalization constraint $C_{ts}(x_s) := \mu_s(x_s) - \sum_{x_t} \mu_{st}(x_s, x_t) = 0$.

Define $N := \{\mu \geq 0 \mid \sum_{x_s} \mu_s(x_s) = 1, \sum_{x_s, x_t} \mu_{st}(x_s, x_t) = 1\}$

Max-product and Linear Programming on Trees

Proposition 8.2 [MarJor08]. Consider the dual function $\mathcal{Q}$ defined by the following partial Lagrangian formulation of the tree-structured LP:

$\mathcal{Q}(\lambda) := \max_{\mu \in N} L(\mu; \lambda)$ where

$$L(\mu; \lambda) := \langle \theta, \mu \rangle + \sum_{(s,t) \in E} [\sum_{x_s} \lambda_{ts}(x_s) C_{ts}(x_s) + \sum_{x_t} \lambda_{st}(x_t) C_{st}(x_t)]$$

For any fixed point M^* of the max-product message passing updates, the vector $\lambda^* := \log M^*$ is an optimal solution of the dual problem $\min_{\lambda} \mathcal{Q}(\lambda)$.

MAP and LP Relaxation

For a general graph $G = (V, E)$, $\mathcal{M}(G) \subseteq \mathcal{L}(G)$, so

$$\begin{aligned}\max_x \sum_{v \in V} \theta_v(x_v) + \sum_{(u,v) \in E} \theta_{uv}(x_u, x_v) &= \max_{\mu \in \mathcal{M}(G)} \langle \theta, \mu \rangle \\ &\leq \max_{\mu \in \mathcal{L}(G)} \langle \theta, \mu \rangle\end{aligned}$$

Solving the maximization LP gives an upper bound on the MAP problem.

Message Passing for MAP LP-Relaxation

LP is in the complexity class of P so efficient algorithms exist to solve them. However, when there is an underlying graph structure in the case of the MAP LP-Relaxation, variants of max-product message passing can be used instead.

Max Product Linear Programming (MPLP) Algorithm

Some new notations (to be consistent with those in the paper [MPLP]):

MAPLPR: $\mu^* = \arg \max_{\mu \in \mathcal{M}_L(G)} \mu \cdot \theta$

where $\mathcal{M}_L(G) = \{\mu \geq 0 \mid \sum_{x_i} \mu_{ij}(x_i, x_j) = \mu_j(x_j), \sum_{x_j} \mu_{ij}(x_i, x_j) = \mu_i(x_i), \forall (i,j) \in E; \sum_{x_i} \mu_i(x_i) = 1, \forall i \in V\}$

MPLP Algorithm: Dual Program

The dual of MAPLPR:
DMAPLPR: $\min \sum_i \max_{x_i} \sum_{k \in N(i)} \max_{x_k} \beta_{ki}(x_k, x_i)$
s.t. $\beta_{ji}(x_j, x_i) + \beta_{ij}(x_i, x_j) = \theta_{ij}(x_i, x_j)$
where the dual variables are $\beta_{ij}(x_i, x_j)$ for all $(i, j), (j, i) \in E$ and values of x_i and x_j.

Derivation follows from the previous approach, see the paper Appendix for details.

MPLP Algorithm: the Algorithm

Inputs: A graph $G = (V, E)$, potential functions $\theta_{ij}(x_i, x_j)$ for each edge $ij \in E$.

Initialization: Initialize messages to any value.

Algorithm:

- Iterate until a stopping criterion is satisfied:
  - Max-product: Iterate over messages and update (c_{ji} shifts the max to zero)

$$m_{ji}(x_i) \leftarrow \max_{x_j} \left[m_j^{-i}(x_j) + \theta_{ij}(x_i, x_j) \right] - c_{ji}$$

  - EMPLP: For each $ij \in E$, update $\lambda_{ji}(x_i)$ and $\lambda_{ij}(x_j)$ simultaneously (the update for $\lambda_{ij}(x_j)$ is the same with i and j exchanged)

$$\lambda_{ji}(x_i) \leftarrow -\frac{1}{2}\lambda_i^{-j}(x_i) + \frac{1}{2}\max_{x_j} \left[\lambda_j^{-i}(x_j) + \theta_{ij}(x_i, x_j) \right]$$

  - NMPLP: Iterate over nodes $i \in V$ and update all $\gamma_{ij}(x_j)$ where $j \in N(i)$

$$\gamma_{ij}(x_j) \leftarrow \max_{x_i} \left[\theta_{ij}(x_i, x_j) - \gamma_{ji}(x_i) + \frac{2}{|N(i)| + 1} \sum_{k \in N(i)} \gamma_{ki}(x_i) \right]$$

- Calculate node "beliefs": Set $b_i(x_i)$ to be the sum of incoming messages into node $i \in V$ (e.g., for NMPLP set $b_i(x_i) = \sum_{k \in N(i)} \gamma_{ki}(x_i)$).

Output: Return assignment $\boldsymbol{x}$ defined as $x_i = \arg\max_{\hat{x}_i} b(\hat{x}_i)$.

Figure 1: The max-product, EMPLP and NMPLP algorithms. Max-product, EMPLP and NMPLP use messages m_{ij}, λ_{ij} and γ_{ij} respectively. We use the notation $m_j^{-i}(x_j) = \sum_{k \in N(j) \setminus i} m_{kj}(x_j)$.

MPLP Algorithm: Understanding the Algorithm

– EMPLP: For each $ij \in E$, update $\lambda_{ji}(x_i)$ and $\lambda_{ij}(x_j)$ simultaneously (the update for $\lambda_{ij}(x_j)$ is the same with i and j exchanged)

$$\lambda_{ji}(x_i) \leftarrow -\frac{1}{2}\lambda_i^{-j}(x_i) + \frac{1}{2}\max_{x_j}\left[\lambda_j^{-i}(x_j) + \theta_{ij}(x_i, x_j)\right]$$

where $\lambda_i^{-j}(x_i) = \sum_{k \in N(i) \setminus j} \lambda_{ki}(x_i)$ and $\lambda_{ki}(x_i) = \max_{x_k} \beta_{ki}(x_k, x_i)$.

EMPLP is essentially performing block coordinate descent on the dual problem.

MPLP Algorithm: Block Coordinate Descent

DMAPLPR: $\min \sum_i \max_{x_i} \sum_{k \in N(i)} \max_{x_k} \beta_{ki}(x_k, x_i)$
s.t. $\beta_{ji}(x_j, x_i) + \beta_{ij}(x_i, x_j) = \theta_{ij}(x_i, x_j)$
where the dual variables are $\beta_{ij}(x_i, x_j)$ for all $(i, j), (j, i) \in E$ and values of x_i and x_j.

Consider fixing all the β variables except the ones corresponding to some edge $(i, j) \in E$ (i.e. β_{ij}, β_{ji})
Rewrite the objective as $f(\beta_{ij}, \beta_{ji}) =$
$\max_{x_i}[\lambda_i^{-j}(x_i) + \max_{x_j} \beta_{ji}(x_j, x_i)] + \max_{x_i}[\lambda_j^{-i}(x_j) + \max_{x_i} \beta_{ij}(x_i, x_j)]$
New goal: minimize $f(\beta_{ij}, \beta_{ji})$ subject to constraints
$\beta_{ij}(x_j, x_i) + \beta_{ij}(x_i, x_j) = \theta_{ij}(x_i, x_j)$.

MPLP Algorithm: Block Coordinate Descent

Proposition 2: Maximizing the function $f(\beta_{ij}, \beta_{ji})$ yields the following $\lambda_{ji}(x_i)$ (and the equivalent expression for $\lambda_{ij}(x_j)$)

$$\lambda_{ji}(x_i) = -\tfrac{1}{2}\lambda_i^{-j}(x_i) + \tfrac{1}{2}\max_{x_j}[\lambda_j^{-i}(x_j) + \theta_{ij}(x_i, x_j)]$$

Proof: rewrite

$$\begin{aligned} f(\beta_{ij}, \beta_{ji}) &= \max_{x_i, x_j}[\lambda_i^{-j}(x_i) + \beta_{ji}(x_j, x_i)] + \max_{x_i, x_j}[\lambda_j^{-i}(x_j) + \beta_{ij}(x_i, x_j)] \\ &\geq \max_{x_i, x_j}[\lambda_i^{-j}(x_i) + \beta_{ji}(x_j, x_i) + \lambda_j^{-i}(x_j) + \beta_{ij}(x_i, x_j)] \\ &= \max_{x_i, x_j}[\lambda_i^{-j}(x_i) + \lambda_j^{-i}(x_j) + \theta_{ij}(x_i, x_j)] \end{aligned}$$

MPLP Algorithm: Block Coordinate Descent

One equality condition:
$\lambda_j^{-i}(x_j) + \beta_{ij}(x_i, x_j) = \lambda_i^{-j}(x_i) + \beta_{ji}(x_j, x_i) =$
$\frac{1}{2}(\theta_{ij}(x_i, x_j) + \lambda_i^{-j}(x_i) + \lambda_j^{-i}(x_j))$, which implies that
$\beta_{ij}(x_i, x_j) = \frac{1}{2}(\theta_{ij}(x_i, x_j) + \lambda_i^{-j}(x_i) - \lambda_j^{-i}(x_j))$ and a similar expression for β_{ji}.
Then
$\lambda_{ij}(x_j) = \max_{x_i} \beta_{ij}(x_i, x_j) = -\frac{1}{2}\lambda_j^{-i}(x_j) + \frac{1}{2}\max_{x_i}[\lambda_i^{-j}(x_i) + \theta_{ij}(x_i, x_j)]$
and similar for $\lambda_{ji}(x_i)$. $\blacksquare$

MPLP Algorithm: Properties

- Convergence guarantee: decrease the dual objective (an upper bound on the MAP value) at every iteration.
- Limit fixed point:
  - if each node belief (aggregate of messages) has a unique maximizer, then LP relaxation is tight.
  - if x_i are binary, the fixed point can be used to obtain the primal optimum.

Conclusion

- MAP inference can be relaxed to an LP problem.
- For tree-structured graphs, the LP-relaxation is tight.
- Connections between max-product message passing and LP-relaxation:
  - For tree-structured graphs, max-product message passing solves the dual of the LP-relaxation.
  - For general graphs, MPLP performs block coordinate descent on the dual of the LP-relaxation and provides convergence guarantees.