Probabilistic Graphical Models

Lecture 9: MAP Inference

Matthew Walter

TTI-Chicago

May 5, 2020

MAP Inference

- Thus far, we have focused on inference in the context of conditional probability queries, i.e., $P(\boldsymbol{Y} \mid \boldsymbol{E} = \boldsymbol{e})$

MAP Inference

- Thus far, we have focused on inference in the context of conditional probability queries, i.e., $P(\boldsymbol{Y} \mid \boldsymbol{E} = \boldsymbol{e})$
- Often, we are instead interested in the most likely assignment to non-evidence variables (e.g., speech recognition or channel decoding)

$$\begin{aligned}
\boldsymbol{y}^* &= \arg\max_{\boldsymbol{y}} P(\boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{E} = \boldsymbol{e}) \\
&= \arg\max_{\boldsymbol{y}} \frac{P(\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{E} = \boldsymbol{e})}{P(\boldsymbol{E} = \boldsymbol{e})} \\
&= \arg\max_{\boldsymbol{y}} P(\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{E} = \boldsymbol{e})
\end{aligned}$$

MAP Inference

- Thus far, we have focused on inference in the context of conditional probability queries, i.e., $P(\boldsymbol{Y} \mid \boldsymbol{E} = \boldsymbol{e})$
- Often, we are instead interested in the most likely assignment to non-evidence variables (e.g., speech recognition or channel decoding)

$$
\begin{aligned}
\boldsymbol{y}^* &= \arg\max_{\boldsymbol{y}} P(\boldsymbol{Y} = \boldsymbol{y} \mid \boldsymbol{E} = \boldsymbol{e}) \\
&= \arg\max_{\boldsymbol{y}} \frac{P(\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{E} = \boldsymbol{e})}{P(\boldsymbol{E} = \boldsymbol{e})} \\
&= \arg\max_{\boldsymbol{y}} P(\boldsymbol{Y} = \boldsymbol{y}, \boldsymbol{E} = \boldsymbol{e})
\end{aligned}
$$

- This is a max over all factors consistent with $\boldsymbol{E} = \boldsymbol{e}$ and thus, a function of $\boldsymbol{E}$ (and a factor itself)
- Map inference is an optimization problem (e.g., energy minimization of negative (log-)likelihood)

MAP Inference: Computational Complexity

- The following decision problem (BN-MAP-DP) is NP-Complete:

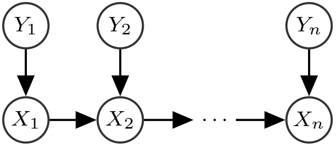
  Given a Bayesian network $\mathcal{B}$ over $\mathcal{X}$ and a number τ, the **BN-MAP-DP** problem is to decide whether an assignment $\boldsymbol{x}$ to $\mathcal{X}$ exists such that $P(\boldsymbol{x}) > \tau$

- The following decision problem is NP-Hard:

  Given a polytree Bayesian network $\mathcal{B}$ over $\mathcal{X}$, a subset $\boldsymbol{Y} \subset \mathcal{X}$, and a number τ, decide whether there exists an assignment $\boldsymbol{Y} = \boldsymbol{y}$ such that $P(\boldsymbol{y}) > \tau$

$$\boldsymbol{y}^* = \underset{Y_1,\ldots,Y_n}{\arg\max} \sum_{X_1,\ldots,X_n} P(Y_1,\ldots,Y_n,X_1,\ldots,X_n)$$

Results in a factor that is exponential in n



MAP Inference

- Consider MAP inference in the context of a Gibbs factorization

$$\arg\max_{\boldsymbol{x}} P(\boldsymbol{x}), \quad \text{where } P(\boldsymbol{x}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \phi_c(\boldsymbol{x}_c)$$

MAP Inference

- Consider MAP inference in the context of a Gibbs factorization

$$\arg\max_{\boldsymbol{x}} P(\boldsymbol{x}), \quad \text{where } P(\boldsymbol{x}) = \frac{1}{Z}\prod_{c\in\mathcal{C}} \phi_c(\boldsymbol{x}_c)$$

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

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

- This is called the **max-product** inference task

MAP Inference

- Consider MAP inference in the context of a Gibbs factorization

$$\arg\max_{\boldsymbol{x}} P(\boldsymbol{x}), \quad \text{where } P(\boldsymbol{x}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \phi_c(\boldsymbol{x}_c)$$

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

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

- This is called the **max-product** inference task
- In log-space: $\theta_c(\boldsymbol{x}_t) = \log \phi_c(\boldsymbol{x}_c)$, this is equivalent to **max-sum** (or min-sum if we negate the terms as a form of energy minimization)

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

MAP Inference

- Consider MAP inference in the context of a Gibbs factorization

$$\arg\max_{\boldsymbol{x}} P(\boldsymbol{x}), \quad \text{where } P(\boldsymbol{x}) = \frac{1}{Z}\prod_{c \in \mathcal{C}} \phi_c(\boldsymbol{x}_c)$$

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

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

- This is called the **max-product** inference task
- In log-space: $\theta_c(\boldsymbol{x}_t) = \log \phi_c(\boldsymbol{x}_c)$, this is equivalent to **max-sum** (or min-sum if we negate the terms as a form of energy minimization)

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

- In general, this is an NP-hard problem

Marginal MAP Inference

- In some cases, we may be interested in MAP inference over a subset of the variables
- This requires marginalizing out the other variables

$$\begin{aligned}
\boldsymbol{x}^* &= \arg\max_{\boldsymbol{x}} P(\boldsymbol{x}) \\
&= \arg\max_{\boldsymbol{x}} \sum_{\boldsymbol{z}} P(\boldsymbol{x}, \boldsymbol{z}) \\
&= \arg\max_{\boldsymbol{x}} \sum_{\boldsymbol{z}} \prod_{c \in \mathcal{C}} \phi_c(\boldsymbol{x}_c)
\end{aligned}$$

- Involves a max, a sum, and a product (even harder)

MAP Inference

- Compare sum-product and max-product (equivalently max-sum in log space) problems:

$$\text{sum-product} \qquad \sum_{\boldsymbol{x}} \prod_{c \in \mathcal{C}} \phi_c(\boldsymbol{x}_c)$$

$$\text{max-sum} \qquad \max_{\boldsymbol{x}} \sum_{c \in \mathcal{C}} \theta_c(\boldsymbol{x}_c)$$

- We can exchange operators $(+, \times)$ for $(\max, +)$ and, because both satisfy associativity and commutativity, we can apply variable elimination and belief propagation
- Gives rise to "max-product variable elimination" and "max-product belief propagation"

MAP Inference: Example

- Consider a simple chain $A - B - C - D$, and suppose that we want to find the MAP assignment

$$\max_{a,b,c,d} \phi_{AB}(a,b)\phi_{BC}(b,c)\phi_{CD}(c,d)$$

MAP Inference: Example

- Consider a simple chain $A - B - C - D$, and suppose that we want to find the MAP assignment

$$\max_{a,b,c,d} \phi_{AB}(a,b)\phi_{BC}(b,c)\phi_{CD}(c,d)$$

- Just as with sum-product VE, we can push maximizations inside

$$\max_{a,b} \phi_{AB}(a,b) \cdot \max_{c} \phi_{BC}(b,c) \cdot \max_{d} \phi_{CD}(c,d)$$

or equivalently for max-sum,

$$\max_{a,b} \theta_{AB}(a,b) + \max_{c} \theta_{BC}(b,c) + \max_{d} \theta_{CD}(c,d)$$

MAP Inference: Example

- Consider a simple chain $A - B - C - D$, and suppose that we want to find the MAP assignment

$$\max_{a,b,c,d} \phi_{AB}(a,b)\phi_{BC}(b,c)\phi_{CD}(c,d)$$

- Just as with sum-product VE, we can push maximizations inside

$$\max_{a,b} \phi_{AB}(a,b) \cdot \overbrace{\max_{c} \phi_{BC}(b,c)}^{\psi(b)} \cdot \overbrace{\max_{d} \phi_{CD}(c,d)}^{\psi(c)}$$

or equivalently for max-sum,

$$\max_{a,b} \theta_{AB}(a,b) + \overbrace{\max_{c} \theta_{BC}(b,c)}^{\psi(b)} + \overbrace{\max_{d} \theta_{CD}(c,d)}^{\psi(c)}$$

- Just as with sum-product VE, each max results in a new factor

MAP Inference: Example

- Consider a simple chain $A - B - C - D$, and suppose that we want to find the MAP assignment

$$\max_{a,b,c,d} \phi_{AB}(a,b)\phi_{BC}(b,c)\phi_{CD}(c,d)$$

- Just as with sum-product VE, we can push maximizations inside

$$\max_{a,b} \phi_{AB}(a,b) \cdot \overbrace{\max_{c} \phi_{BC}(b,c)}^{\psi(b)} \cdot \overbrace{\max_{d} \phi_{CD}(c,d)}^{\psi(c)}$$

or equivalently for max-sum,

$$\max_{a,b} \theta_{AB}(a,b) + \overbrace{\max_{c} \theta_{BC}(b,c)}^{\psi(b)} + \overbrace{\max_{d} \theta_{CD}(c,d)}^{\psi(c)}$$

- Just as with sum-product VE, each max results in a new factor
- We find the actual maximizing assignment by *traceback*

Max Marginals

- Consider a function $f : \text{Val}(\boldsymbol{X}) \rightarrow \mathbb{R}$
- The **max-marginal** of a function f relative to variables $\boldsymbol{Y} \subseteq \boldsymbol{X}$ is

$$\text{MaxMarg}_f(\boldsymbol{y}) = \max_{\xi\langle \boldsymbol{Y} \rangle = \boldsymbol{y}} f(\xi)$$

(where $\xi\langle \boldsymbol{Y} \rangle = \boldsymbol{y}$ restricts ξ to subsets for which $\boldsymbol{Y} = \boldsymbol{y}$)

Max Marginals

- Consider a function $f : \mathrm{Val}(\boldsymbol{X}) \rightarrow \mathbb{R}$
- The **max-marginal** of a function f relative to variables $\boldsymbol{Y} \subseteq \boldsymbol{X}$ is

$$\mathrm{MaxMarg}_f(\boldsymbol{y}) = \max_{\xi\langle \boldsymbol{Y} \rangle = \boldsymbol{y}} f(\xi)$$

(where $\xi\langle \boldsymbol{Y} \rangle = \boldsymbol{y}$ restricts ξ to subsets for which $\boldsymbol{Y} = \boldsymbol{y}$)

- For example, if $f = \tilde{P}_\Phi$, then $\mathrm{MaxMarg}_{\tilde{P}_\Phi}(\boldsymbol{y})$ is the (unnormalized) probability of the most likely assignment consistent with $\boldsymbol{Y} = \boldsymbol{y}$

Max Marginals

- Consider a function f : Val$(\boldsymbol{X}) \rightarrow \mathbb{R}$
- The **max-marginal** of a function f relative to variables $\boldsymbol{Y} \subseteq \boldsymbol{X}$ is

$$\text{MaxMarg}_f(\boldsymbol{y}) = \max_{\xi\langle\boldsymbol{Y}\rangle=\boldsymbol{y}} f(\xi)$$

(where $\xi\langle\boldsymbol{Y}\rangle = \boldsymbol{y}$ restricts ξ to subsets for which $\boldsymbol{Y} = \boldsymbol{y}$)
- For example, if $f = \tilde{P}_\Phi$, then $\text{MaxMarg}_{\tilde{P}_\Phi}(\boldsymbol{y})$ is the (unnormalized) probability of the most likely assignment consistent with $\boldsymbol{Y} = \boldsymbol{y}$
- Consider the Bayesian network: $A \rightarrow B$

$$\begin{aligned}\max_{a,b} P(a,b) &= \max_{a,b} P(a)P(b \mid a)\\ &= \max_a P(a) \max_b P(b \mid a)\\ &= \max_a P(a)\phi(a)\end{aligned}$$

Max Marginals

- Consider a function $f : \mathsf{Val}(\boldsymbol{X}) \rightarrow \mathbb{R}$
- The **max-marginal** of a function f relative to variables $\boldsymbol{Y} \subseteq \boldsymbol{X}$ is

$$\mathsf{MaxMarg}_f(\boldsymbol{y}) = \max_{\xi\langle \boldsymbol{Y} \rangle = \boldsymbol{y}} f(\xi)$$

  (where $\xi\langle \boldsymbol{Y} \rangle = \boldsymbol{y}$ restricts ξ to subsets for which $\boldsymbol{Y} = \boldsymbol{y}$)
- For example, if $f = \tilde{P}_\Phi$, then $\mathsf{MaxMarg}_{\tilde{P}_\Phi}(\boldsymbol{y})$ is the (unnormalized) probability of the most likely assignment consistent with $\boldsymbol{Y} = \boldsymbol{y}$
- Consider the Bayesian network: $A \rightarrow B$

$$\begin{aligned}
\max_{a,b} P(a, b) &= \max_{a,b} P(a) P(b \,|\, a) \\
&= \max_{a} P(a) \max_{b} P(b \,|\, a) \\
&= \max_{a} P(a) \phi(a)
\end{aligned}$$

- Just like sum-product, MAP becomes an operation on factors

Factor Maximization

- Given a set of variables $\boldsymbol{X}$ and $Y \notin \boldsymbol{X}$ with a factor $\phi(\boldsymbol{X}, Y)$. The **factor maximization** of Y in ϕ is a new factor $\psi(\boldsymbol{X})$:

$$\psi(\boldsymbol{X}) = \max_{Y} \phi(\boldsymbol{X}, Y)$$

- For example, $\psi(A, C) = \max_B \phi(A, B, C)$ is a factor maximization

a^1	b^1	c^1	0.25
a^1	b^1	c^2	0.35
a^1	b^2	c^1	0.08
a^1	b^2	c^2	0.16
a^2	b^1	c^1	0.05
a^2	b^1	c^2	0.07
a^2	b^2	c^1	0
a^2	b^2	c^2	0
a^3	b^1	c^1	0.15
a^3	b^1	c^2	0.21
a^3	b^2	c^1	0.09
a^3	b^2	c^2	0.18

a^1	c^1	0.25
a^1	c^2	0.35
a^2	c^1	0.05
a^2	c^2	0.07
a^3	c^1	0.15
a^3	c^2	0.21

- Gives rise to **max-product variable elimination**
  - Replace sum with max
  - Traceback to recover the most likely sequence

Max-Product Variable Elimination

Algorithm 13.1 Variable elimination algorithm for MAP. The algorithm can be used both in its max-product form, as shown, or in its max-sum form, replacing factor product with factor addition.

Procedure Max-Product-VE (
  Φ, // Set of factors over $\boldsymbol{X}$
  $\prec$ // Ordering on $\boldsymbol{X}$
)
  Let $X_1, \ldots, X_k$ be an ordering of $\boldsymbol{X}$ such that
    $X_i \prec X_j$ iff $i < j$
  for $i = 1, \ldots, k$
    $(\Phi, \phi_{X_i}) \leftarrow$ Max-Product-Eliminate-Var(Φ, X_i)
  $\boldsymbol{x}^* \leftarrow$ Traceback-MAP$(\{\phi_{X_i} \;:\; i = 1, \ldots, k\})$
  return $\boldsymbol{x}^*, \Phi$ // Φ contains the probability of the MAP

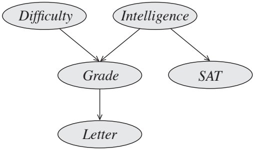
Procedure Max-Product-Eliminate-Var (
  Φ, // Set of factors
  Z // Variable to be eliminated
)
  $\Phi' \leftarrow \{\phi \in \Phi \;:\; Z \in \mathit{Scope}[\phi]\}$
  $\Phi'' \leftarrow \Phi - \Phi'$
  $\psi \leftarrow \prod_{\phi \in \Phi'} \phi$
  $\tau \leftarrow \max_Z \psi$
  return $(\Phi'' \cup \{\tau\}, \psi)$

Procedure Traceback-MAP (
  $\{\phi_{X_i} \;:\; i = 1, \ldots, k\}$
)
  for $i = k, \ldots, 1$
    $\boldsymbol{u}_i \leftarrow (x^*_{i+1}, \ldots, x^*_k)\langle \mathit{Scope}[\phi_{X_i}] - \{X_i\}\rangle$
      // The maximizing assignment to the variables eliminated after X_i
    $x^*_i \leftarrow \arg\max_{x_i} \phi_{X_i}(x_i, \boldsymbol{u}_i)$
      // x^*_i is chosen so as to maximize the corresponding entry in the factor, relative to the previous choices $\boldsymbol{u}_i$
  return $\boldsymbol{x}^*$

Max-Product Variable Elimination

- Let's revisit a simplified version of the Student example from the text

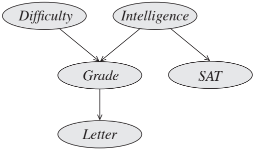
$$\arg\max_{S,I,D,L,G} P(S, I, D, L, G)$$



Max-Product Variable Elimination

- Let's revisit a simplified version of the Student example from the text

$$\arg\max_{S,I,D,L,G} P(S, I, D, L, G)$$



- Max-product variable elimination proceeds as

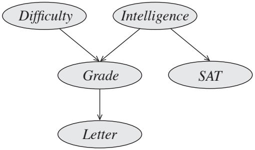
Step	Variable eliminated	Factors used	Intermediate factor	New factor
1	S	$\phi_S(I, S)$	$\psi_1(I, S)$	$\tau_1(I)$
2	I	$\phi_I(I)$, $\phi_G(G, I, D)$, $\tau_1(I)$	$\psi_2(G, I, D)$	$\tau_2(G, D)$
3	D	$\phi_D(D)$, $\tau_2(G, D)$	$\psi_3(G, D)$	$\tau_3(G)$
4	L	$\phi_L(L, G)$	$\psi_4(L, G)$	$\tau_4(G)$
5	G	$\tau_4(G)$, $\tau_3(G)$	$\psi_5(G)$	$\tau_5(\emptyset)$

- The evaluation of the computational complexity of max-product VE is identical to that of sum-product VE

Max-Product Variable Elimination

- Let's revisit a simplified version of the Student example from the text

$$\arg\max_{S,I,D,L,G} P(S,I,D,L,G)$$



- Max-product variable elimination proceeds as

Step	Variable eliminated	Factors used	Intermediate factor	New factor
1	S	$\phi_S(I,S)$	$\psi_1(I,S)$	$\tau_1(I)$
2	I	$\phi_I(I)$, $\phi_G(G,I,D)$, $\tau_1(I)$	$\psi_2(G,I,D)$	$\tau_2(G,D)$
3	D	$\phi_D(D)$, $\tau_2(G,D)$	$\psi_3(G,D)$	$\tau_3(G)$
4	L	$\phi_L(L,G)$	$\psi_4(L,G)$	$\tau_4(G)$
5	G	$\tau_4(G)$, $\tau_3(G)$	$\psi_5(G)$	$\tau_5(\emptyset)$

- The evaluation of the computational complexity of max-product VE is identical to that of sum-product VE
- But, how do we recover $\{S,I,D,L,G\}^* = \arg\max_{S,I,D,L,G} P(S,I,D,L,G)$?

Decoding

- We are interested in **decoding**: Finding the most likely assignment
- As we eliminated variables, we can only determine their "conditional" maximizing value as a function of non-eliminated variables
- The result of max-product variable elimination is the max-marginal over the last variable X_i, $\text{MaxMarg}_{\tilde{P}_\Phi}(X_i)$
- This corresponds to the probability of the most likely assignments consistent with $X_i = x_i$
- We can select the most likely assignment x_i^* and work backwards (i.e., "trace it back"), without needing to revisit the decision
- This process is referred to as **traceback**
- Traceback is linear in the number of variables (i.e., no added expense)

Variable Elimination for Marginal MAP

$$
\begin{aligned}
\boldsymbol{y}^* &= \arg\max_{\boldsymbol{y}} P(\boldsymbol{y}) \\
&= \arg\max_{\boldsymbol{y}} \sum_{\boldsymbol{z}} P(\boldsymbol{y}, \boldsymbol{z}) \\
&= \arg\max_{\boldsymbol{y}} \sum_{\boldsymbol{z}} \prod_{c \in \mathcal{C}} \phi_c(\boldsymbol{x}_c)
\end{aligned}
$$

- Similarly involves summations and maximizations over factors
- Might suggest that we can use the same tricks as before
- However, max and sum operations do not commute and we have to perform all summations before maximizations
- Constrains the legal variable elimination orderings
- Use sum-product algorithm to marginalize out $\boldsymbol{X} \backslash \boldsymbol{Y}$
- Use max-product to maximize over $\boldsymbol{Y}$

Max-product Belief Propagation in Clique Trees

- Suppose that we have a clique tree $\mathcal{T}$ with cliques $\boldsymbol{C}_1, \ldots \boldsymbol{C}_k$
- Max-product belief propagation proceeds just like sum-product BP, just with different messages

$$m_{i \to j}(\boldsymbol{S}_{i,j}) = \max_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \left\{ \psi_i(\boldsymbol{C}_i) \cdot \prod_{k \in \mathsf{Nb}_i - \{j\}} m_{k \to i} \right\}$$

- Upon calibration, we have the single node max-marginals

$$\beta_i(\boldsymbol{c}_i) = \mathsf{MaxMarg}_{\tilde{P}_\Phi}(\boldsymbol{c}_i)$$

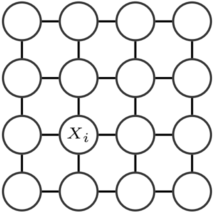
- Similar to sum-product BP, the tree is **max-calibrated** when

$$\max_{\boldsymbol{C}_i - \boldsymbol{S}_{i,j}} \beta_i = \max_{\boldsymbol{C}_j - \boldsymbol{S}_{i,j}} \beta_j = \mu_{i,j}(\boldsymbol{S}_{i,j})$$

- If the MAP assignment $\boldsymbol{x}^*$ is **unique** (i.e., no ties), we can find it by locally decoding each max-marginal

$$x_i^* = \arg\max_{x_i} \mathsf{MaxMarg}_{\tilde{P}_\Phi}(X_i)$$

Exact MAP Inference Beyond Clique Trees



- Consider an MRF with unary and pairwise potentials

$$P(x_1, \ldots, x_n) = \frac{1}{Z} \prod_{i,j} \phi_{ij}(x_i, x_j) \cdot \prod_i \phi_i(x_i)$$

- The treewidth (optimal induced width) of an $n \times n$ Ising model is n
- Thus, exact (MAP) inference is exponential in n, which is intractable for practical values of n

Exact MAP Inference Beyond Clique Trees

- MAP inference corresponds to a discrete optimization problem

$$\arg\max_{\boldsymbol{x}} \sum_{i \in V} \theta_i(x_i) + \sum_{c \in \mathcal{C}} \theta_c(\boldsymbol{x}_c)$$

  where we include unary terms without loss of generality
- As framed, this is a general discrete optimization problem, which can be used to express many hard combinatorial optimization problems (e.g., 3-SAT)

Dual Decomposition

- Consider the MAP problem for a pairwise Markov random field

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

- MAP is a hard (i.e., combinatorial) optimization problem

Dual Decomposition

- Consider the MAP problem for a pairwise Markov random field

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

- MAP is a hard (i.e., combinatorial) optimization problem
- Pushing the maximizations inside the sums only increases the value

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

- Unlike MAP, the optimizations on the righthand side are easy

Dual Decomposition

- Consider the MAP problem for a pairwise Markov random field

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

- MAP is a hard (i.e., combinatorial) optimization problem
- Pushing the maximizations inside the sums only increases the value

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

- Unlike MAP, the optimizations on the righthand side are easy
- Key insight: Construct a *dual* function $L(\delta) \geq \text{MAP}(\theta)$, and solve for δ that minimizes $L(\delta)$

Dual Decomposition

- Consider the following reparameterization

$$\tilde{\theta}_i(x_i) = \theta_i(x_i) + \sum_{ij \in E} \delta_{j \to i}(x_i)$$

$$\tilde{\theta}_{ij}(x_i, x_j) = \theta_{ij}(x_i, x_j) - \delta_{j \to i}(x_i) - \delta_{i \to j}(x_j)$$

Dual Decomposition

- Consider the following reparameterization

$$\tilde{\theta}_i(x_i) = \theta_i(x_i) + \sum_{ij \in E} \delta_{j \to i}(x_i)$$

$$\tilde{\theta}_{ij}(x_i, x_j) = \theta_{ij}(x_i, x_j) - \delta_{j \to i}(x_i) - \delta_{i \to j}(x_j)$$

- It is easy to verify that

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

Dual Decomposition

- Consider the following reparameterization

$$\tilde{\theta}_i(x_i) = \theta_i(x_i) + \sum_{ij \in E} \delta_{j \to i}(x_i)$$

$$\tilde{\theta}_{ij}(x_i, x_j) = \theta_{ij}(x_i, x_j) - \delta_{j \to i}(x_i) - \delta_{i \to j}(x_j)$$

- It is easy to verify that

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

- Thus, we have

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

Dual Decomposition

- Consider the following reparameterization

$$\tilde{\theta}_i(x_i) = \theta_i(x_i) + \sum_{ij \in E} \delta_{j \to i}(x_i)$$

$$\tilde{\theta}_{ij}(x_i, x_j) = \theta_{ij}(x_i, x_j) - \delta_{j \to i}(x_i) - \delta_{i \to j}(x_j)$$

- It is easy to verify that

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

- Thus, we have

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

- Each value of δ gives a different upper bound on the MAP value
- We get the **tightest** upper bound by minimizing the RHS with respect to δ

Dual Decomposition: Derivation

- Consider a general factorization over unary potentials and factors F

$$\mathsf{MAP}(\boldsymbol{\theta}) = \max_{\boldsymbol{x}} \sum_{i \in V} \theta_i(x_i) + \sum_{f \in F} \theta_f(\boldsymbol{x}_f)$$

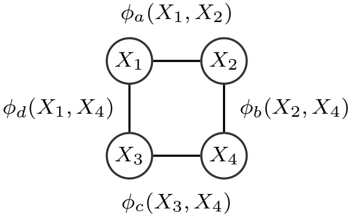
- Let x_i^f be the copy of x_i used by factor f, and let $\boldsymbol{x}_f^f = \{x_i^f\}_{i \in f}$ be the set of variables used by factor f
- We can express the optimization problem equivalently as

$$\begin{aligned} &\max \sum_{i \in V} \theta_i(x_i) + \sum_{f \in F} \theta_f(\boldsymbol{x}_f^f) \\ &\text{s.t.} \quad x_i^f = x_i \quad \forall f, i \in f \end{aligned}$$

- Without the constraints, we would have independent maximizations over each factor, an easy optimization problem

Dual Decomposition: Derivation

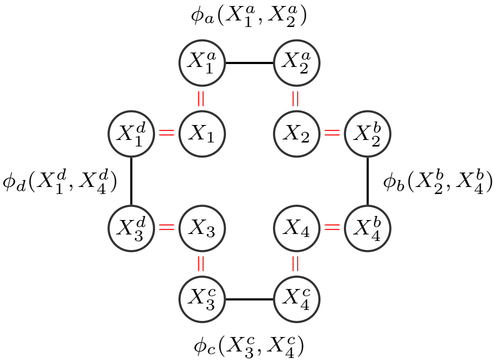
Consider a simple pairwise MRF with unary and pairwise factors



$$\text{MAP}(\boldsymbol{\theta}) = \max_{\boldsymbol{x}} \sum_{i \in V} \theta_i(x_i) + \sum_{f \in F} \theta_f(\boldsymbol{x}_f)$$

Dual Decomposition: Derivation

We can formulate the objective as a constrained optimization over individual factors



$$\max \sum_{i \in V} \theta_i(x_i) + \sum_{f \in F} \theta_f(\boldsymbol{x}_f^f)$$
$$\text{s.t.} \quad x_i^f = x_i \quad \forall f, i \in f$$

The need to maintain agreement among factors is what makes the problem difficult

Dual Decomposition: Derivation

- We can reformulate this optimization via Lagrangian relaxation

$$
\begin{aligned}
L(\boldsymbol{\delta}, \boldsymbol{x}, \boldsymbol{x}^F) = & \sum_{i \in V} \theta_i(x_i) + \sum_{f \in F} \theta_f(\boldsymbol{x}_f^f) \\
& + \sum_{f \in F} \sum_{i \in f} \sum_{\hat{x}_i} \delta_{fi}(\hat{x}_i) \left(\mathbb{1}[x_i = \hat{x}_i] - \mathbb{1}[x_i^f = \hat{x}_i] \right)
\end{aligned}
$$

where $\boldsymbol{\delta} = \{\delta_{fi}(x_i) : f \in F, i \in f, x_i\}$ are Lagrange multipliers and $L(\boldsymbol{\delta}, \boldsymbol{x}, \boldsymbol{x}^F)$ is the *Lagrangian*

Dual Decomposition: Derivation

- We can reformulate this optimization via Lagrangian relaxation

$$
\begin{aligned}
L(\boldsymbol{\delta}, \boldsymbol{x}, \boldsymbol{x}^F) = & \sum_{i \in V} \theta_i(x_i) + \sum_{f \in F} \theta_f(\boldsymbol{x}_f^f) \\
& + \sum_{f \in F} \sum_{i \in f} \sum_{\hat{x}_i} \delta_{fi}(\hat{x}_i) \left(\mathbb{1}[x_i = \hat{x}_i] - \mathbb{1}[x_i^f = \hat{x}_i] \right)
\end{aligned}
$$

  where $\boldsymbol{\delta} = \{\delta_{fi}(x_i) : f \in F, i \in f, x_i\}$ are Lagrange multipliers and $L(\boldsymbol{\delta}, \boldsymbol{x}, \boldsymbol{x}^F)$ is the *Lagrangian*

- The following is equivalent to that on the previous slide

$$
\begin{aligned}
& \max_{\boldsymbol{x}, \boldsymbol{x}^F} L(\boldsymbol{\delta}, \boldsymbol{x}, \boldsymbol{x}^F) \\
& \text{s.t.} \quad x_i^f = x_i \quad \forall f, i \in f
\end{aligned}
$$

Dual Decomposition: Derivation

- To make this problem tractable, ignore the equality constraints

$$L(\boldsymbol{\delta}) = \max_{\boldsymbol{x},\boldsymbol{x}^F} L(\boldsymbol{\delta}, \boldsymbol{x}, \boldsymbol{x}^F)$$

$$= \sum_{i \in V} \max_{x_i} \left(\theta_i(x_i) + \sum_{f:i \in f} \delta_{fi}(x_i) \right) + \sum_{f \in F} \max_{\boldsymbol{x}_f^f} \left(\theta_f(\boldsymbol{x}_f^f) - \sum_{i \in f} \delta_{fi}(x_i^f) \right)$$

Dual Decomposition: Derivation

- To make this problem tractable, ignore the equality constraints

$$
\begin{aligned}
L(\boldsymbol{\delta}) &= \max_{\boldsymbol{x}, \boldsymbol{x}^F} L(\boldsymbol{\delta}, \boldsymbol{x}, \boldsymbol{x}^F) \\
&= \sum_{i \in V} \max_{x_i} \left(\theta_i(x_i) + \sum_{f : i \in f} \delta_{fi}(x_i) \right) + \sum_{f \in F} \max_{\boldsymbol{x}_f^f} \left(\theta_f(\boldsymbol{x}_f^f) - \sum_{i \in f} \delta_{fi}(x_i^f) \right)
\end{aligned}
$$

- $L(\boldsymbol{\delta})$ maximizes over a larger space (since $\boldsymbol{x}$ and $\boldsymbol{x}^F$ may differ)
- Consequently, MAP$(\boldsymbol{\theta}) \leq L(\boldsymbol{\delta})$

Dual Decomposition: Derivation

- To make this problem tractable, ignore the equality constraints

$$
\begin{aligned}
L(\boldsymbol{\delta}) &= \max_{\boldsymbol{x},\boldsymbol{x}^F} L(\boldsymbol{\delta},\boldsymbol{x},\boldsymbol{x}^F) \\
&= \sum_{i\in V} \max_{x_i} \left(\theta_i(x_i) + \sum_{f:i\in f} \delta_{fi}(x_i) \right) + \sum_{f\in F} \max_{\boldsymbol{x}_f^f} \left(\theta_f(\boldsymbol{x}_f^f) - \sum_{i\in f} \delta_{fi}(x_i^f) \right)
\end{aligned}
$$

- $L(\boldsymbol{\delta})$ maximizes over a larger space (since $\boldsymbol{x}$ and $\boldsymbol{x}^F$ may differ)
- Consequently, $\text{MAP}(\boldsymbol{\theta}) \leq L(\boldsymbol{\delta})$
- The *dual problem* is to find the tightest upper bound by optimizing the Lagrange multipliers

$$\min_{\boldsymbol{\delta}} L(\boldsymbol{\delta})$$

Dual Decomposition: Reparameterization

- As we saw earlier, we can reparameterize the parameters $\boldsymbol{\theta}$ as $\tilde{\boldsymbol{\theta}}$:

$$\tilde{\theta}_i^{\boldsymbol{\delta}}(x_i) = \theta_a(x_i) + \sum_{f:i \in f} \delta_{fi}(x_i)$$

$$\tilde{\theta}_f^{\boldsymbol{\delta}}(\boldsymbol{x}_f) = \theta_f(\boldsymbol{x}_f) - \sum_{i \in f} \delta_{fi}(x_i)$$

where $\boldsymbol{\delta}$ is a set of dual variables

Dual Decomposition: Reparameterization

- As we saw earlier, we can reparameterize the parameters $\boldsymbol{\theta}$ as $\tilde{\boldsymbol{\theta}}$:

$$\tilde{\theta}_i^{\boldsymbol{\delta}}(x_i) = \theta_a(x_i) + \sum_{f:i\in f} \delta_{fi}(x_i)$$

$$\tilde{\theta}_f^{\boldsymbol{\delta}}(\boldsymbol{x}_f) = \theta_f(\boldsymbol{x}_f) - \sum_{i\in f} \delta_{fi}(x_i)$$

where $\boldsymbol{\delta}$ is a set of dual variables

- We can then rewrite $L(\boldsymbol{\delta})$ as

$$L(\boldsymbol{\delta}) = \sum_{i\in V} \max_{x_i} \left(\theta_i(x_i) + \sum_{f:i\in f} \delta_{fi}(x_i) \right) + \sum_{f\in F} \max_{\boldsymbol{x}_f^f} \left(\theta_f(\boldsymbol{x}_f^f) - \sum_{i\in f} \delta_{fi}(x_i^f) \right)$$

$$= \sum_{i\in V} \max_{x_i} \tilde{\theta}_i^{\boldsymbol{\delta}}(x_i) + \sum_{f\in F} \max_{\boldsymbol{x}_f} \tilde{\theta}_f^{\boldsymbol{\delta}}(\boldsymbol{x}_f)$$

Dual Decomposition

- Let's revisit the pairwise Markov random field example from Slide 17
- The Lagrangian becomes

$$
\begin{aligned}
L(\boldsymbol{\delta}) = \sum_{i \in V} \max_{x_i} \left(\theta_i(x_i) + \sum_{ij \in E} \delta_{j \to i}(x_i) \right) \\
+ \max_{x_i, x_j} \left(\theta_{ij}(x_i, x_j) - \delta_{j \to i}(x_i) - \delta_{i \to j}(x_j) \right)
\end{aligned}
$$

- This results in the dual objective

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

- This provides an upper-bound on the MAP assignment

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

- How do we find a setting for $\boldsymbol{\delta}$ that provides tight bounds?

Solving the Dual Objective

- $L(\boldsymbol{\delta})$ is convex and continuous, but is non-differentiable at points $\boldsymbol{\delta}$ where $\tilde{\theta}_i^{\boldsymbol{\delta}}(x_i)$ and $\tilde{\theta}_f^{\boldsymbol{\delta}}(\boldsymbol{x}_f)$ have multiple optima
- There are many methods for optimizing non-differentiable objectives
  1. Subgradient algorithms
  2. Block coordinate descent algorithms

Solving the Dual Objective: Subgradient Methods[1]

- A *subgradient* of $L(\boldsymbol{\delta})$ is a vector $g_{\boldsymbol{\delta}}$ such that

$$L(\boldsymbol{\delta}') \geq L(\boldsymbol{\delta}) + g_{\boldsymbol{\delta}} \cdot (\boldsymbol{\delta}' - \boldsymbol{\delta}) \;\text{ for all } \boldsymbol{\delta}'$$

- A subgradient approach alternates between computing the subgradients (maximizing subproblems) and updating the dual parameters $\boldsymbol{\delta}$ via the subgradients
- An iteration of the subgradient descent procedure follows as

$$\delta_{fi}^{t+1}(x_i) = \delta_{fi}^{t}(x_i) - \alpha_t g_{fi}^{t}(x_i)$$

  where $\boldsymbol{g}^t$ is a subgradient of $L(\boldsymbol{\delta})$ and α_t is a step-size
- Converges when $\lim_{t \rightarrow \infty} \alpha_t = 0$ and $\sum_{t=0}^{\infty} \alpha_t = \infty$ (e.g., $\alpha_t = \frac{1}{t}$)

[1]See Komodakis et al., "MRF energy minimization and beyond via dual decomposition," PAMI 2010

Solving the Dual Objective: Block Coordinate Descent

- Basic idea: Fix all but a set of dual variables and then minimize the objective with respect to this set
- Questions:
  1. Which set of variables do we update?
  2. How do we update these variables?
- Coordinate descent methods are local, easy to implement, and converge faster than subgradient methods
- However, there are cases in which they do not converge

Solving the Dual Objective: MPLP Algorithm

The Max Product Linear Programming (MPLP) algorithm is a coordinate descent algorithm that fixes all $\boldsymbol{\delta}$ except $\delta_{fi}(x_i)$ for a specific f and for all i

Algorithm Max Product Linear Programming Algorithm

Require: $\{\theta_i(x_i), \theta_{ij}(x_i, x_j)\}\ \forall i, j$

Initialize $\delta_{i \to j}(x_j) = 0, \delta_{j \to i}(x_i) = 0 \quad \forall i, j \in E, x_i, x_j$

while change in $L(\boldsymbol{\delta})$ is not small enough **do**

    for each edge $i, j \in E$ **do**

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

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

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

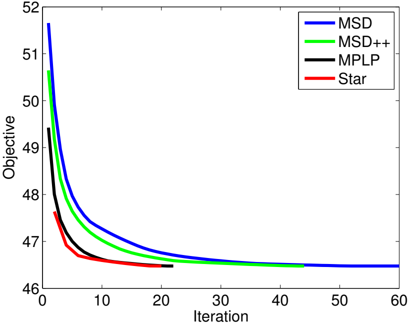
    end for

end while

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

Dual Decomposition: Experimental Results

A comparison of different block coordinate descent algorithms on a 10×10 Ising model

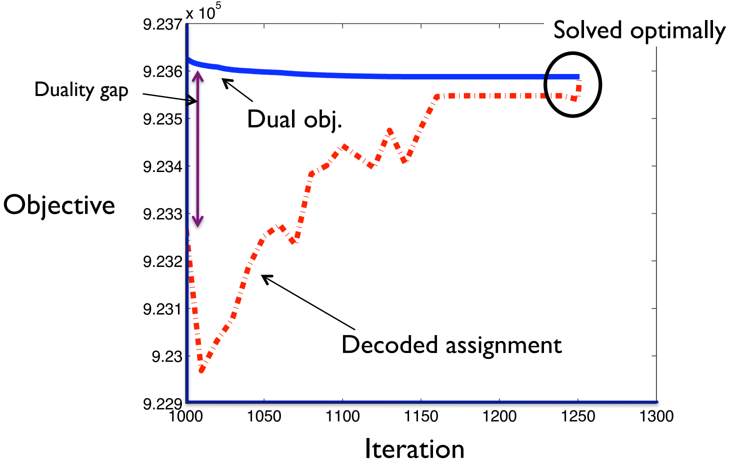


(Courtesy Sontag et al., 2011)

MSD	Node-adjacent updates
MPLP	Edge updates
Star	Updates on all incident edges

Dual Decomposition: Experimental Results

An evaluation on a stereo vision inference task



(Courtesy Sontag et al., 2011)