Visual Recognition: Inference in Graphical Models

Raquel Urtasun

TTI Chicago

Feb 23, 2012

Graphical Models

Graphical models

- Applications
- Representation
- Inference
  - message passing (LP relaxations)
  - graph cuts
- Learning

Classification algorithms

We want to classify an object $x \in \mathcal{X}$ into labels $y \in \mathcal{Y}$

- First there was binary $y \in \{-1, 1\}$

$x \rightarrow$  ; $y \rightarrow \{table, notable\}$

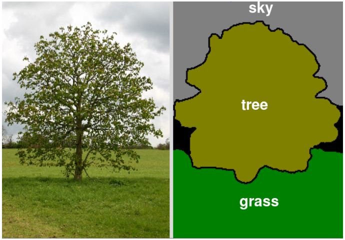
- Then multiclass $y \in \{1, \cdots, \mathcal{C}\}$

$x \rightarrow$  ; $y \rightarrow \{car, bus, bicycle\}$

- The next generation is structured labels

Structure Prediction Problems

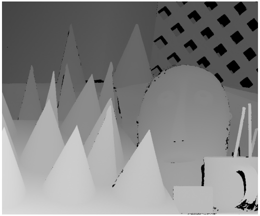
- Segmentation and detection



- Stereo

Structure Prediction Problems

- Segmentation and detection
- Stereo



- 3D scene understanding

Structure Prediction Problems

- Segmentation and detection
- Stereo
- 3D scene understanding

- Multi-labeling of images

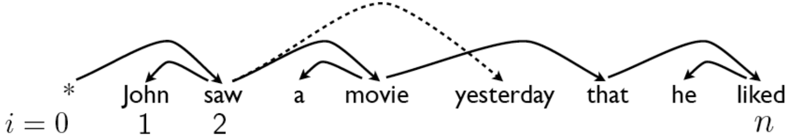
Structure Prediction Problems

- Segmentation and detection
- Stereo
- 3D scene understanding
- Multi-labeling of images

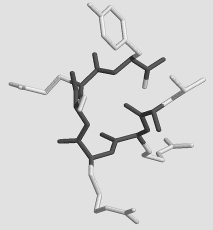
$man - made, vehicle, car.$

Structure Prediction Problems

- Segmentation and detection
- Stereo
- 3D scene understanding
- Multi-labeling of images
- Other fields, e.g., part of speech tagging, parsing, protein folding.



MRLLILALLGICSLTAYIVEGVGSEVSDKR
TCVSLTTQRLPVSRIKTYTITEGSLRAVIF
ITKRGLKVCADPQATWVRDVVRSMDRKSNT
RNNMIQTKPTGTQQSTNTAVTLTG



Why structured?

- Independent prediction is good but...

- Neighboring pixels should have same labels (if they look similar).

Why structured?

- Independent prediction is good but...

- Neighboring pixels should have same labels (if they look similar).

- Learning and inference is tractable for tree-shaped models or binary variables with submodular energies.

Why structured?

- Independent prediction is good but...

- Neighboring pixels should have same labels (if they look similar).

- Learning and inference is tractable for tree-shaped models or binary variables with submodular energies.

Structure Prediction

- Input: $x \in \mathcal{X}$, typically an image.
- Output: label $y \in \mathcal{Y}$.
- Consider a score function $\theta(x, y)$ called **potential** or **feature** such that

$$\theta(x, y) = \begin{cases} \text{high} & \text{if } y \text{ is a good label for } x \\ \text{low} & \text{if } y \text{ is a bad label for } x \end{cases}$$

- We want to predict a label as

$$y^* = \arg\max_y \theta(x, y)$$

Score Decomposition

- We assume that the score decomposes

$$\theta(y|x) = \sum_i \theta_i(y_i) + \sum_\alpha \theta_\alpha(y_\alpha)$$

- This represents a (conditional) Markov Random Field (CRF)

$$p(x, y) = \frac{1}{Z} \prod_i \psi_i(x, y_i) \prod_\alpha \psi_\alpha(x, y_\alpha)$$

with $\log \psi_i(x, y_i) = \theta_i(x, y_i)$, and $\log \psi_\alpha(x, y_\alpha) = \theta_\alpha(x, y_\alpha)$.

Score Decomposition

- We assume that the score decomposes

$$\theta(y|x) = \sum_i \theta_i(y_i) + \sum_\alpha \theta_\alpha(y_\alpha)$$

- This represents a (conditional) Markov Random Field (CRF)

$$p(x, y) = \frac{1}{Z} \prod_i \psi_i(x, y_i) \prod_\alpha \psi_\alpha(x, y_\alpha)$$

with $\log \psi_i(x, y_i) = \theta_i(x, y_i)$, and $\log \psi_\alpha(x, y_\alpha) = \theta_\alpha(x, y_\alpha)$.

- $Z = \sum_y \prod_i \psi_i(x, y_i) \prod_\alpha \psi_\alpha(x, y_\alpha)$ is the partition function.

Score Decomposition

- We assume that the score decomposes

$$\theta(y|x) = \sum_i \theta_i(y_i) + \sum_\alpha \theta_\alpha(y_\alpha)$$

- This represents a (conditional) Markov Random Field (CRF)

$$p(x, y) = \frac{1}{Z} \prod_i \psi_i(x, y_i) \prod_\alpha \psi_\alpha(x, y_\alpha)$$

with $\log \psi_i(x, y_i) = \theta_i(x, y_i)$, and $\log \psi_\alpha(x, y_\alpha) = \theta_\alpha(x, y_\alpha)$.

- $Z = \sum_y \prod_i \psi_i(x, y_i) \prod_\alpha \psi_\alpha(x, y_\alpha)$ is the partition function.
- Prediction also decomposes

$$y^* = \arg\max_y \sum_i \theta_i(y_i) + \sum_\alpha \theta_\alpha(y_\alpha)$$

Score Decomposition

- We assume that the score decomposes

$$\theta(y|x) = \sum_i \theta_i(y_i) + \sum_\alpha \theta_\alpha(y_\alpha)$$

- This represents a (conditional) Markov Random Field (CRF)

$$p(x, y) = \frac{1}{Z} \prod_i \psi_i(x, y_i) \prod_\alpha \psi_\alpha(x, y_\alpha)$$

  with $\log \psi_i(x, y_i) = \theta_i(x, y_i)$, and $\log \psi_\alpha(x, y_\alpha) = \theta_\alpha(x, y_\alpha)$.

- $Z = \sum_y \prod_i \psi_i(x, y_i) \prod_\alpha \psi_\alpha(x, y_\alpha)$ is the partition function.
- Prediction also decomposes

$$y^* = \arg\max_y \sum_i \theta_i(y_i) + \sum_\alpha \theta_\alpha(y_\alpha)$$

- This in general is NP hard.

Score Decomposition

- We assume that the score decomposes

$$\theta(y|x) = \sum_i \theta_i(y_i) + \sum_\alpha \theta_\alpha(y_\alpha)$$

- This represents a (conditional) Markov Random Field (CRF)

$$p(x, y) = \frac{1}{Z} \prod_i \psi_i(x, y_i) \prod_\alpha \psi_\alpha(x, y_\alpha)$$

  with $\log \psi_i(x, y_i) = \theta_i(x, y_i)$, and $\log \psi_\alpha(x, y_\alpha) = \theta_\alpha(x, y_\alpha)$.
- $Z = \sum_y \prod_i \psi_i(x, y_i) \prod_\alpha \psi_\alpha(x, y_\alpha)$ is the partition function.
- Prediction also decomposes

$$y^* = \arg\max_y \sum_i \theta_i(y_i) + \sum_\alpha \theta_\alpha(y_\alpha)$$

- This in general is NP hard.

Markov Networks

- A **clique** in an undirected graph is a subset of its vertices such that every two vertices in the subset are connected by an edge.
- A **maximal clique** is a clique that cannot be extended by including one more adjacent vertex.

Markov Networks

- A **clique** in an undirected graph is a subset of its vertices such that every two vertices in the subset are connected by an edge.
- A **maximal clique** is a clique that cannot be extended by including one more adjacent vertex.
- For a set of variables $\mathbf{y} = \{y_1, \cdots, y_N\}$ a **Markov network** is defined as a product of potentials over the maximal cliques y_α of the graph G

$$p(y_1, \cdots, y_N) = \frac{1}{Z} \prod_\alpha \psi_\alpha(y_\alpha)$$

Markov Networks

- A **clique** in an undirected graph is a subset of its vertices such that every two vertices in the subset are connected by an edge.
- A **maximal clique** is a clique that cannot be extended by including one more adjacent vertex.
- For a set of variables $\mathbf{y} = \{y_1, \cdots, y_N\}$ a **Markov network** is defined as a product of potentials over the maximal cliques y_α of the graph G

$$p(y_1, \cdots, y_N) = \frac{1}{Z} \prod_\alpha \psi_\alpha(y_\alpha)$$

- Special case: cliques of size 2 – pairwise Markov network

Markov Networks

- A **clique** in an undirected graph is a subset of its vertices such that every two vertices in the subset are connected by an edge.
- A **maximal clique** is a clique that cannot be extended by including one more adjacent vertex.
- For a set of variables $\mathbf{y} = \{y_1, \cdots, y_N\}$ a **Markov network** is defined as a product of potentials over the maximal cliques y_α of the graph G

$$p(y_1, \cdots, y_N) = \frac{1}{Z} \prod_\alpha \psi_\alpha(y_\alpha)$$

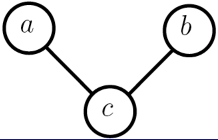
- Special case: cliques of size 2 – pairwise Markov network
- In case all potentials are strictly positive this is called a Gibbs distribution

Markov Networks

- A **clique** in an undirected graph is a subset of its vertices such that every two vertices in the subset are connected by an edge.
- A **maximal clique** is a clique that cannot be extended by including one more adjacent vertex.
- For a set of variables $\mathbf{y} = \{y_1, \cdots, y_N\}$ a **Markov network** is defined as a product of potentials over the maximal cliques y_α of the graph G

$$p(y_1, \cdots, y_N) = \frac{1}{Z} \prod_\alpha \psi_\alpha(y_\alpha)$$

- Special case: cliques of size 2 – pairwise Markov network
- In case all potentials are strictly positive this is called a Gibbs distribution
- Example: $p(a, b, c) = \frac{1}{Z} \psi_{a,c}(a, c) \psi_{bc}(b, c)$

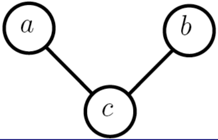


Markov Networks

- A **clique** in an undirected graph is a subset of its vertices such that every two vertices in the subset are connected by an edge.
- A **maximal clique** is a clique that cannot be extended by including one more adjacent vertex.
- For a set of variables $\mathbf{y} = \{y_1, \cdots, y_N\}$ a **Markov network** is defined as a product of potentials over the maximal cliques y_α of the graph G

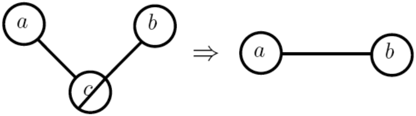
$$p(y_1, \cdots, y_N) = \frac{1}{Z} \prod_\alpha \psi_\alpha(y_\alpha)$$

- Special case: cliques of size 2 – pairwise Markov network
- In case all potentials are strictly positive this is called a Gibbs distribution
- Example: $p(a, b, c) = \frac{1}{Z} \psi_{a,c}(a, c) \psi_{bc}(b, c)$

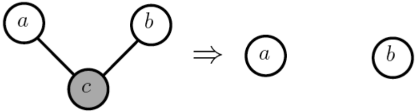


Properties of Markov Network

- Marginalizing over c makes *a* and *b* dependent



- Conditioning on *c* makes *a* and *b* independent



[Source: P. Gehler]

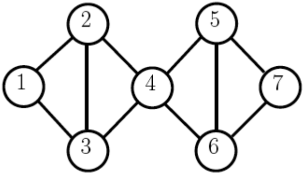
Local and Global Markov properties

- **Local Markov property**: condition on neighbours makes indep. of the rest

$$p(y_i|\mathbf{y} \setminus \{y_i\}) = p(y|ne(y_i))$$

  Example: $y_4 \perp \{y_1, y_7\} | \{y_2, y_3, y_5, y_6\}$

- **Global Markov Property**: For disjoint sets of variables $(\mathcal{A}, \mathcal{B}, \mathcal{S})$, where $\mathcal{S}$ separates $\mathcal{A}$ from $\mathcal{B}$ then $\mathcal{A} \perp \mathcal{B} | \mathcal{S}$
- $\mathcal{S}$ is called a separator.
- Example: $y_1 \perp y_7 | \{y_4\}$



[Source: P. Gehler]

Relationship Potentials to Graphs

- Consider

$$p(a,b,c) = \frac{1}{Z}\psi(a,b)\psi(b,c)\psi(c,a)$$

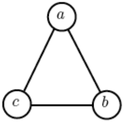
- What is the corresponding Markov network (graphical representation)?

Relationship Potentials to Graphs

- Consider

$$p(a, b, c) = \frac{1}{Z}\psi(a, b)\psi(b, c)\psi(c, a)$$

- What is the corresponding Markov network (graphical representation)?



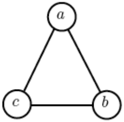
- Which other factorization is represented by this network?

Relationship Potentials to Graphs

- Consider

$$p(a, b, c) = \frac{1}{Z}\psi(a, b)\psi(b, c)\psi(c, a)$$

- What is the corresponding Markov network (graphical representation)?



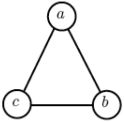
- Which other factorization is represented by this network?

Relationship Potentials to Graphs

- Consider

$$p(a,b,c) = \frac{1}{Z}\psi(a,b)\psi(b,c)\psi(c,a)$$

- What is the corresponding Markov network (graphical representation)?



- Which other factorization is represented by this network?

$$p(a,b,c) = \frac{1}{Z}\psi(a,b,c)$$

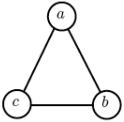
- The factorization is not specified by the graph

Relationship Potentials to Graphs

- Consider

$$p(a,b,c) = \frac{1}{Z}\psi(a,b)\psi(b,c)\psi(c,a)$$

- What is the corresponding Markov network (graphical representation)?



- Which other factorization is represented by this network?

$$p(a,b,c) = \frac{1}{Z}\psi(a,b,c)$$

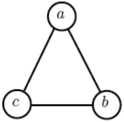
- The factorization is not specified by the graph
- Let's look at Factor Graphs

Relationship Potentials to Graphs

- Consider

$$p(a, b, c) = \frac{1}{Z}\psi(a, b)\psi(b, c)\psi(c, a)$$

- What is the corresponding Markov network (graphical representation)?



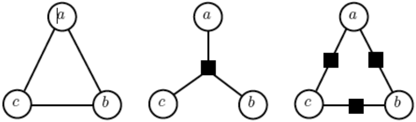
- Which other factorization is represented by this network?

$$p(a, b, c) = \frac{1}{Z}\psi(a, b, c)$$

- The factorization is not specified by the graph
- Let's look at Factor Graphs

Factor Graphs

Now consider we introduce an extra node (a square) for each factor



(a) (b) (c)

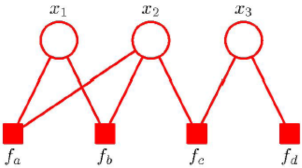
The **factor graph (FG)** has a node (represented by a square) for each factor $\psi(y_\alpha)$ and a variable node (represented by a circle) for each variable x_i.

- Left: Markov Network
- Middle: Factor graph representation of $\psi(a, b, c)$
- Right: Factor graph representation of $\psi(a, b)\psi(b, c)\psi(c, a)$
- Different factor graphs can have the same Markov network

[Source: P. Gehler]

Examples

- Which distribution?



- What factor graph?

$$p(x_1, x_2, x_3) = p(x_1)p(x_2)p(x_3|x_1, x_2)$$

[Source: P. Gehler]

Inference in trees

- Given distribution $p(y_1, \cdots, y_n)$
- Inference: computing functions of the distribution
  - mean
  - marginal
  - conditionals
- Marginal inference in singly-connected graph (trees)
- Later: extensions to loopy graphs

[Source: P. Gehler]

Variable Elimination

- Consider Markov chain ($a, b, c, d \in \{0, 1\}$)

  $$a \leftarrow b \leftarrow c \leftarrow d$$

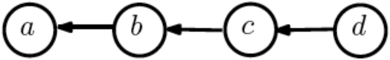
  with distribution

  $$p(a, b, c, d) = p(a \mid b)p(b \mid c)p(c \mid d)p(d)$$

- Task: compute the marginal $p(a)$

[Source: P. Gehler]

Variable Elimination



$$
\begin{aligned}
p(a) &= \sum_{b,c,d} p(a, b, c, d) \\
&= \sum_{b,c,d} p(a \mid b) p(b \mid c) p(c \mid d) p(d)
\end{aligned}
$$

- Naive: $2 \times 2 \times 2 = 8$ states to sum over
- Re-order summation:

$$
p(a) = \sum_{b,c} p(a \mid b) p(b \mid c) \underbrace{\sum_{d} p(c \mid d) p(d)}_{\gamma_d(c)}
$$

[Source: P. Gehler]

Variable Elimination

$$p(a) = \sum_{b,c} p(a \mid b) p(b \mid c) \underbrace{\sum_{d} p(c \mid d) p(d)}_{\gamma_d(c)}$$

$$p(a) = \sum_{b} p(a \mid b) \underbrace{\sum_{c} p(b \mid c) \gamma_d(c)}_{\gamma_c(b)}$$

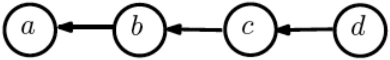
$$p(a) = \sum_{b} p(a \mid b) \gamma_c(b)$$

- We need $2 + 2 + 2 = 6$ calculations
- For a chain of length T scale linearly $n * 2$, cf naive approach 2^n

[Source: P. Gehler]

Finding Conditional Marginals

- Again:



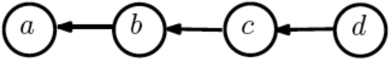
$$p(a, b, c, d) = p(a \mid b)p(b \mid c)p(c \mid d)p(d)$$

- Now find $p(d \mid a)$

$$\begin{aligned} p(d \mid a) &\propto \sum_{b,c} p(a \mid b)p(b \mid c)p(c \mid d)p(d) \\ &= \sum_c \underbrace{\sum_b p(a \mid b)p(b \mid c)}_{\gamma_b(c)} p(c \mid d)p(d) \\ &\stackrel{def}{=} \gamma_c(d) \text{ not a distribution} \end{aligned}$$

[Source: P. Gehler]

Finding Conditional Marginals



- Found that

$$p(d \mid a) = k\gamma_c(d)$$

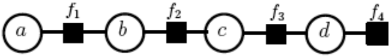
- and since $\sum_d p(d \mid a) = 1$

$$k = \frac{1}{\sum_d \gamma_c(d)}$$

- Again $\gamma_c(d)$ is not a distribution (but a message)

[Source: P. Gehler]

Now with factor graphs



$$p(a,b,c,d) = f_1(a,b)f_2(b,c)f_3(c,d)f_4(d)$$

$$\begin{aligned} p(a,b,c) &= \sum_d p(a,b,c,d) \\ &= f_1(a,b)f_2(b,c)\underbrace{\sum_d f_3(c,d)f_4(d)}_{\mu_{d\to c}(c)} \end{aligned}$$

$$p(a,b) = \sum_c p(a,b,c) = f_1(a,b)\underbrace{\sum_c f_2(b,c)\mu_{d\to c}(c)}_{\mu_{c\to b}(b)}$$

[Source: P. Gehler]

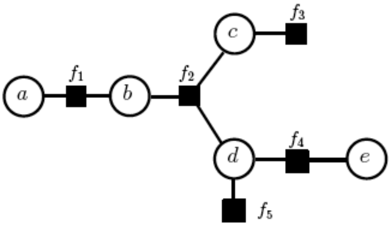
Inference in Chain Structured Factor Graphs

- Simply recurse further
- $\gamma_{m \to n}(n)$ carries the information beyond m
- We did not need the factors in general (next) we will see that making a distinction is helpful

[Source: P. Gehler]

General singly-connected factor graphs I

- Now consider a branching graph:



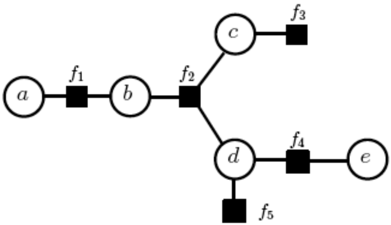
with factors

$$f_1(a, b)f_2(b, c, d)f_3(c)f_4(d, e)f_5(d)$$

- For example: find marginal $p(a, b)$

[Source: P. Gehler]

General singly-connected factor graphs II

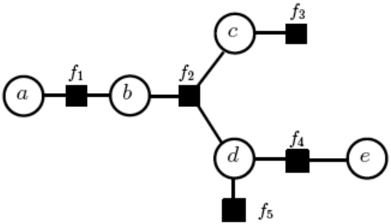


$$p(a,b) = f_1(a,b) \underbrace{\sum_{c,d,e} f_2(b,c,d) f_3(c) f_5(d) f_4(d,e)}_{\mu_{f_2 \to b}(b)}$$

$$\mu_{f_2 \to b}(b) = \sum_{c,d} f_2(b,c,d) \underbrace{f_3(c)}_{\mu_{c \to f_2}(c)} \underbrace{f_5(d) \sum_{e} f_4(d,e)}_{\mu_{d \to f_2}(d)}$$

[Source: P. Gehler]

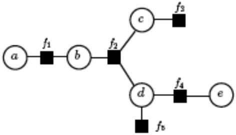
General singly-connected factor graphs III



$$\mu_{d \to f_2}(d) = \underbrace{f_5(d)}_{\mu_{f_5 \to d}(d)} \underbrace{\sum_e f_4(d, e)}_{\mu_{f_4 \to d}(d)}$$

[Source: P. Gehler]

General singly-connected factor graphs IV



- If we want to compute the marginal $p(a)$:

$$p(a) = \underbrace{\sum_b f_1(a,b)\mu_{f_2 \to b}(b)}_{\mu_{f_1 \to a}(a)}$$

- which we could also view as

$$p(a) = \sum_b f_1(a,b)\underbrace{\mu_{f_2 \to b}(b)}_{\mu_{b \to f_1}(b)}$$

[Source: P. Gehler]

Summary

- Once computed, messages can be re-used
- All marginals $p(c), p(d), p(c, d), \cdots$ can be written as a function of messages
- We need an algorithm to compute all messages: Sum-Product algorithm

[Source: P. Gehler]

Sum-product algorithm overview

- Algorithm to compute all messages efficiently, assuming the graph is singly-connected
- It can be used to compute any desired marginals
- Also known as belief propagation (BP)

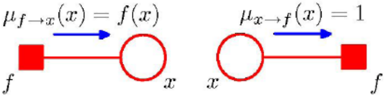
The algorithm is composed of

1. Initialization
2. Variable to Factor message
3. Factor to Variable message

[Source: P. Gehler]

1. Initialization

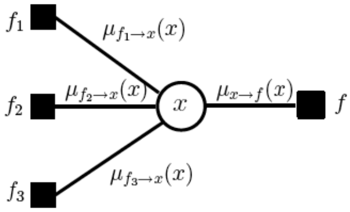
- Messages from extremal (simplical) node factors are initialized to the factor (left)
- Messages from extremal (simplical) variable nodes are set to unity (right)



[Source: P. Gehler]

2. Variable to Factor message

$$\mu_{x \to f}(x) = \prod_{g \in \{\text{ne}(x) \setminus f\}} \mu_{g \to x}(x)$$

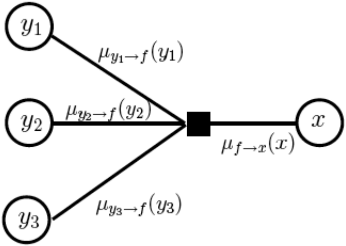


[Source: P. Gehler]

3. Factor to Variable message

- We sum over all states in the set of variables
- This explains the name for the algorithm (sum-product)

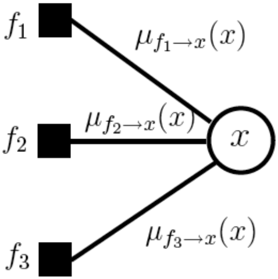
$$\mu_{f\to x}(x) = \sum_{y \in \mathcal{X}_f \setminus x} \phi_f(\mathcal{X}_f) \prod_{y \in \{\mathrm{ne}(f) \setminus x\}} \mu_{y\to f}(y)$$



[Source: P. Gehler]

Marginal computation

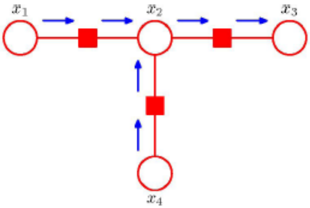
$$p(x) \propto \prod_{f \in \mathrm{ne}(x)} \mu_{f \to x}(x)$$

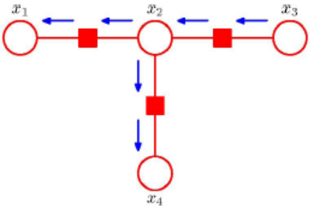


[Source: P. Gehler]

Message Ordering

- Messages depend on previous computed messages
- Only extremal nodes/factors do not depend on other messages
- To compute all messages in the graph
  1. leaf-to-root: (pick root node, compute messages pointing towards root)
  2. root-to-leave: (compute messages pointing away from root)

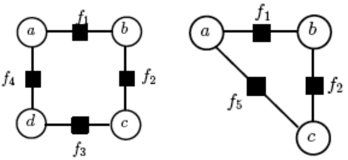




[Source: P. Gehler]

Problems with loops

- Marginalizing over d introduces new link (changes graph structure – in contrast to singly connected graphs)



$$p(a, b, c, d) = f_1(a, b) f_2(b, c) f_3(c, d) f_4(d, a)$$

and marginal

$$p(a, b, c) = f_1(a, b) f_2(b, c) \underbrace{\sum_d f_3(c, d) f_4(d, a)}_{f_5(a,c)}$$

[Source: P. Gehler]

What to infer?

- Mean

$$\mathbb{E}_{p(x)}[x] = \sum_{x \in \mathcal{X}} x p(x)$$

- Mode

$$x^* = \operatorname*{argmax}_{x \in \mathcal{X}} p(x)$$

- Conditional Distributions

$$p(x_i, x_j \mid x_k, x_l) \text{or} p(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$$

- Max-Marginals

$$x_i^* = \operatorname*{argmax}_{x_i \in \mathcal{X}_i} p(x_i) = \cdots dx_n \operatorname*{argmax}_{x_i \in \mathcal{X}_i} \int_{(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)} p(x) dx_1$$

[Source: P. Gehler]

Computing the Partition Function

- The partition function ($p(x) = \frac{1}{Z}\prod_f \Phi_f(\mathcal{X}_f)$) (normalization constant) Z can be computed after the leaf-to-root step (no need for the root-to-leaf step) (choose any $x \in \mathcal{X}$)

$$
\begin{aligned}
Z &= \sum_{\mathcal{X}} \prod_f \phi_f(\mathcal{X}_f) && (10)\\
&= \sum_x \sum_{\mathcal{X}\setminus\{x\}} \prod_{f\in \mathrm{ne}(x)} \prod_{f \notin \mathrm{ne}(x)} \phi_f(\mathcal{X}_f) && (11)\\
&= \sum_x \prod_{f\in \mathrm{ne}(x)} \sum_{\mathcal{X}\setminus\{x\}} \prod_{f \notin \mathrm{ne}(x)} \phi_f(\mathcal{X}_f) && (12)\\
&= \sum_x \prod_{f\in \mathrm{ne}(x)} \mu_{f\to x}(x) && (13)
\end{aligned}
$$

[Source: P. Gehler]

Log Messages

- In large graphs, messages may become very small
- Work with log-messages instead $\lambda = \log \mu$
- Variable-to-factor messages

$$\mu_{x \to f}(x) = \prod_{g \in \{\text{ne}(x) \setminus f\}} \mu_{g \to x}(x)$$

then becomes

$$\lambda_{x \to f}(x) = \sum_{g \in \{\text{ne}(x) \setminus f\}} \lambda_{g \to x}(x)$$

[Source: P. Gehler]

Log Messages

- Work with log-messages instead $\lambda = \log \mu$
- Factor-to-Variable messages

$$\mu_{f \to x}(x) = \sum_{y \in \mathcal{X}_f \setminus x} \Phi_f(\mathcal{X}_f) \prod_{y \in \{\mathrm{ne}(f) \setminus x\}} \mu_{y \to f}(y) \tag{16}$$

then becomes

$$\lambda_{f \to x}(x) = \log \left(\sum_{y \in \mathcal{X}_f \setminus x} \Phi(\mathcal{X}_f) \exp \left[\sum_{y \in \{\mathrm{ne}(f) \setminus x\}} \lambda_{y \to f}(y) \right] \right) \tag{17}$$

[Source: P. Gehler]

Trick

- Log-Factor-to-Variable Message:

$$\lambda_{f \to x}(x) = \log \sum_{y \in \mathcal{X}_f \backslash x} \Phi_f(\mathcal{X}_f) \exp \sum_{y \in \{\text{ne}(f) \backslash x\}} \lambda_{y \to f}(y) \quad (18)$$

- large numbers lead to numerical instability
- Use the following equality

$$\log \sum_i \exp(v_i) = \alpha + \log \sum_i \exp(v_i - \alpha) \quad (19)$$

- With $\alpha = \max \lambda_{y \to f}(y)$

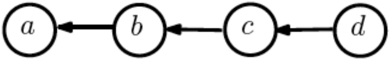
[Source: P. Gehler]

Finding the maximal state: Max-Product

- For a given distribution $p(x)$ find the most likely state:

$$x^* = \underset{x_1,\ldots,x_n}{\operatorname{argmax}}\, p(x_1, \ldots, x_n)$$

- Again use factorization structure to distribute the maximisation to local computations
- Example: chain



$$f(x_1, x_2, x_3, x_4) = \phi(x_1, x_2)\phi(x_2, x_3)\phi(x_3, x_1)$$

[Source: P. Gehler]

Be careful: not maximal marginal states!

- The most likely state

$$x^* = \underset{x_1,\ldots,x_n}{\operatorname{argmax}}\, p(x_1, \ldots, x_n)$$

  does not need to be the one for which the marginals are maximized:

- For all $i = 1, \ldots, n$

$$x_i^* = \underset{x_i}{\operatorname{argmax}}\, p(x_i)$$

- Example:

	$x = 0$	$x = 1$
$y = 0$	0.3	0.4
$y = 1$	0.3	0.0

[Source: P. Gehler]

Example chain

$$
\begin{aligned}
\max_x f(x) &= \max_{x_1,x_2,x_3,x_4} \phi(x_1,x_2)\phi(x_2,x_3)\phi(x_3,x_4) \\
&= \max_{x_1,x_2,x_3} \phi(x_1,x_2)\phi(x_2,x_3) \underbrace{\max_{x_4} \phi(x_3,x_4)}_{\gamma(x_3)} \\
&= \max_{x_1,x_2} \phi(x_1,x_2) \underbrace{\max_{x_3} \phi(x_2,x_3)\gamma(x_3)}_{\gamma(x_2)} \\
&= \max_{x_1} \underbrace{\max_{x_2} \phi(x_1,x_2)\gamma(x_2)}_{\gamma(x_1)} \\
&= \max_{x_1} \gamma(x_1)
\end{aligned}
$$

[Source: P. Gehler]

Example chain

- Once computed the messages ($\gamma(\cdot)$) find the optimal values

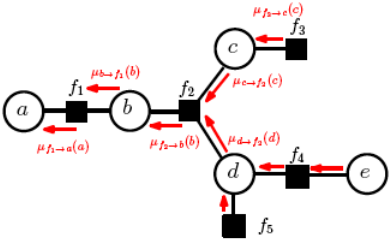
$$
\begin{aligned}
x_1^* &= \operatorname*{argmax}_{x_1} \gamma(x_1) \\
x_2^* &= \operatorname*{argmax}_{x_2} \phi(x_1^*, x_2)\gamma(x_2) \\
x_3^* &= \operatorname*{argmax}_{x_3} \phi(x_2^*, x_3)\gamma(x_3) \\
x_4^* &= \operatorname*{argmax}_{x_4} \phi(x_3^*, x_4)\gamma(x_4)
\end{aligned}
$$

- this is called backtracking (an application of dynamic programming)
- can choose arbitrary start point

[Source: P. Gehler]

Trees

- Spot the messages:



$$
\begin{aligned}
\max_x f(x) &= \max_{a,b,c,d,e} f_1(a,b) f_2(b,c,d) f_3(c) f_4(d,e) f_5(d) \\
&= \max_a \mu_{f_2 \to a}(a)
\end{aligned}
$$

[Source: P. Gehler]

Max-Product Algorithm

Pick any variable as root and

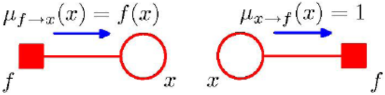
1. Initialisation (same as sum-product)
2. Variable to Factor message (same as sum-product)
3. Factor to Variable message

Then compute the maximal state

[Source: P. Gehler]

1. Initialization

- Messages from extremal node factors are initialized to the factor
- Messages from extremal variable nodes are set to unity



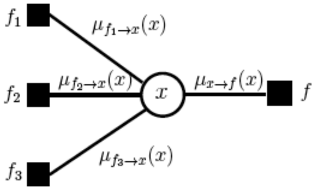
- Same as sum product

[Source: P. Gehler]

2. Variable to Factor message

- Same as for sum-product

$$\mu_{x \to f}(x) = \prod_{g \in \{\mathrm{ne}(x) \setminus f\}} \mu_{g \to x}(x)$$

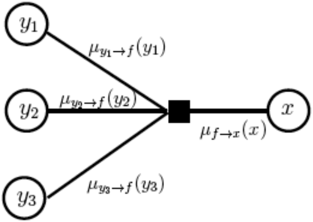


[Source: P. Gehler]

3. Factor to Variable message

- Different message than in sum-product
- This is now a max-product

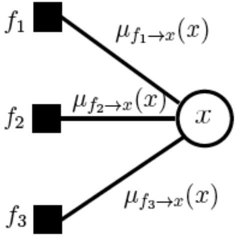
$$\mu_{f \to x}(x) = \max_{y \in \mathcal{X}_f \setminus x} \phi_f(\mathcal{X}_f) \prod_{y \in \{\text{ne}(f) \setminus x\}} \mu_{y \to f}(y)$$



[Source: P. Gehler]

Maximal state of Variable

$$x^* = \underset{x}{\operatorname{argmax}} \prod_{f \in \mathrm{ne}(x)} \mu_{f \to x}(x)$$



- This does not work with loops
- Same problem as the sum product algorithm

[Source: P. Gehler]

Inference with LP relaxations

Decomposition Solvers

- Solving the problem is hard, as it involves exponential many labels

$$\max_{y_1,\cdots,y_n} \sum_i \theta_i(y_i) + \sum_\alpha \theta_\alpha(y_\alpha)$$

- Trivial decomposition upper bound

$$\sum_i \max_{y_i} \theta_i(y_i) + \sum_\alpha \max_{y_\alpha} \theta_\alpha(y_\alpha)$$

Decomposition Solvers

- Solving the problem is hard, as it involves exponential many labels

$$\max_{y_1,\cdots,y_n} \sum_i \theta_i(y_i) + \sum_\alpha \theta_\alpha(y_\alpha)$$

- Trivial decomposition upper bound

$$\sum_i \max_{y_i} \theta_i(y_i) + \sum_\alpha \max_{y_\alpha} \theta_\alpha(y_\alpha)$$

- Solves MAP in trivial cases, e.g., $\theta_\alpha = 0$.

Decomposition Solvers

- Solving the problem is hard, as it involves exponential many labels

$$\max_{y_1,\cdots,y_n} \sum_i \theta_i(y_i) + \sum_\alpha \theta_\alpha(y_\alpha)$$

- Trivial decomposition upper bound

$$\sum_i \max_{y_i} \theta_i(y_i) + \sum_\alpha \max_{y_\alpha} \theta_\alpha(y_\alpha)$$

- Solves MAP in trivial cases, e.g., $\theta_\alpha = 0$.
- Fails for example

$$\theta_1(y_1) = \theta_2(y_2) = \begin{bmatrix} 3 \\ 0 \end{bmatrix} \qquad \text{max argument} \rightarrow y_1 = y_2 = 0$$

$$\theta_{1,2}(y_1, y_2) = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \qquad \text{max argument} \rightarrow y_1 = y_2 = 1$$

Decomposition Solvers

- Solving the problem is hard, as it involves exponential many labels

$$\max_{y_1,\cdots,y_n} \sum_i \theta_i(y_i) + \sum_\alpha \theta_\alpha(y_\alpha)$$

- Trivial decomposition upper bound

$$\sum_i \max_{y_i} \theta_i(y_i) + \sum_\alpha \max_{y_\alpha} \theta_\alpha(y_\alpha)$$

- Solves MAP in trivial cases, e.g., $\theta_\alpha = 0$.
- Fails for example

$$\theta_1(y_1) = \theta_2(y_2) = \begin{bmatrix} 3 \\ 0 \end{bmatrix} \qquad \text{max argument} \rightarrow y_1 = y_2 = 0$$

$$\theta_{1,2}(y_1, y_2) = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \qquad \text{max argument} \rightarrow y_1 = y_2 = 1$$

- We need to balance non-agreements between arguments

Decomposition Solvers

- Solving the problem is hard, as it involves exponential many labels

$$\max_{y_1,\cdots,y_n} \sum_i \theta_i(y_i) + \sum_\alpha \theta_\alpha(y_\alpha)$$

- Trivial decomposition upper bound

$$\sum_i \max_{y_i} \theta_i(y_i) + \sum_\alpha \max_{y_\alpha} \theta_\alpha(y_\alpha)$$

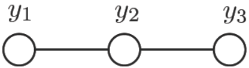
- Solves MAP in trivial cases, e.g., $\theta_\alpha = 0$.
- Fails for example

$$\theta_1(y_1) = \theta_2(y_2) = \begin{bmatrix} 3 \\ 0 \end{bmatrix} \qquad \text{max argument} \rightarrow y_1 = y_2 = 0$$

$$\theta_{1,2}(y_1, y_2) = \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \qquad \text{max argument} \rightarrow y_1 = y_2 = 1$$

- We need to balance non-agreements between arguments

Alternative Representation



$y_i \in \{0, 1\}$

- We want to solve

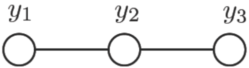
$$\max_{y_1, y_2, y_3} \theta_1(y_1) + \theta_2(y_2) + \theta_3(y_3) + \theta_{12}(y_1, y_2) + \theta_{23}(y_2, y_3)$$

- We can parameterize the problem as

$$\theta_1(y_1) + \theta_2(y_2) + \theta_3(y_3) + \theta_{12}(y_1, y_2) + \theta_{23}(y_2, y_3) = \begin{bmatrix} \theta_1(0) \\ \theta_1(1) \\ \theta_2(0) \\ \theta_2(1) \\ \theta_3(0) \\ \theta_3(1) \\ \theta_{12}(0,0) \\ \theta_{12}(1,0) \\ \theta_{12}(0,1) \\ \theta_{12}(1,1) \\ \theta_{23}(0,0) \\ \theta_{23}(1,0) \\ \theta_{23}(0,1) \\ \theta_{23}(1,1) \end{bmatrix}^T \cdot \begin{bmatrix} b_1(0) \\ b_1(1) \\ b_2(0) \\ b_2(1) \\ b_3(0) \\ b_3(1) \\ b_{12}(0,0) \\ b_{12}(1,0) \\ b_{12}(0,1) \\ b_{12}(1,1) \\ b_{23}(0,0) \\ b_{23}(1,0) \\ b_{23}(0,1) \\ b_{23}(1,1) \end{bmatrix}$$

with **b** satisfying certain conditions, i.e., define the marginal polytope.

Alternative Representation



$$y_i \in \{0, 1\}$$

- We want to solve

$$\max_{y_1, y_2, y_3} \theta_1(y_1) + \theta_2(y_2) + \theta_3(y_3) + \theta_{12}(y_1, y_2) + \theta_{23}(y_2, y_3)$$

- We can parameterize the problem as

$$\theta_1(y_1) + \theta_2(y_2) + \theta_3(y_3) + \theta_{12}(y_1, y_2) + \theta_{23}(y_2, y_3) = \begin{bmatrix} \theta_1(0) \\ \theta_1(1) \\ \theta_2(0) \\ \theta_2(1) \\ \theta_3(0) \\ \theta_3(1) \\ \theta_{12}(0,0) \\ \theta_{12}(1,0) \\ \theta_{12}(0,1) \\ \theta_{12}(1,1) \\ \theta_{23}(0,0) \\ \theta_{23}(1,0) \\ \theta_{23}(0,1) \\ \theta_{23}(1,1) \end{bmatrix}^T \cdot \begin{bmatrix} b_1(0) \\ b_1(1) \\ b_2(0) \\ b_2(1) \\ b_3(0) \\ b_3(1) \\ b_{12}(0,0) \\ b_{12}(1,0) \\ b_{12}(0,1) \\ b_{12}(1,1) \\ b_{23}(0,0) \\ b_{23}(1,0) \\ b_{23}(0,1) \\ b_{23}(1,1) \end{bmatrix}$$

with **b** satisfying certain conditions, i.e., define the marginal polytope.

ILP formulation

- Introduce indicator variables $b_i(y_i)$ for each variable and $b_\alpha(y_\alpha)$ for the factors, and define

$$\max \sum_{\alpha, y_\alpha} b_\alpha(y_\alpha)\theta_\alpha(y_\alpha) + \sum_{i, y_i} b_i(y_i)\theta_i(y_i)$$

subject to:

$$b_i(y_i), b_\alpha(y_\alpha) \in \{0, 1\},$$

$$\sum_{y_\alpha} b_\alpha(y_\alpha) = 1, \sum_{y_i} b_i(y_i) = 1$$

$$\forall i, y_i, \alpha \in N(i), \sum_{y_\alpha \setminus y_i} b_\alpha(y_\alpha) = b_i(y_i)$$

- This ILP is NP-Hard

LP Relaxation

- Replace the integrality constraint

$$\max \sum_{\alpha, y_\alpha} b_\alpha(y_\alpha)\theta_\alpha(y_\alpha) + \sum_{i, y_i} b_i(y_i)\theta_i(y_i)$$

subject to:

$$b_i(y_i), b_\alpha(y_\alpha) \in [0,1], \quad \sum_{y_\alpha} b_\alpha(y_\alpha) = 1, \sum_{y_i} b_i(y_i) = 1$$

$$\forall i, y_i, \alpha \in N(i), \sum_{y_\alpha \backslash y_i} b_\alpha(y_\alpha) = b_i(y_i)$$

- Introduce entropy barrier functions to be smooth and get rid of the simplex constraints

$$\max \sum_{\alpha, y_\alpha} b_\alpha(y_\alpha)\theta_\alpha(y_\alpha) + \sum_{i, y_i} b_i(y_i)\theta_i(y_i) + \epsilon \left(\sum_\alpha c_\alpha H(b_\alpha) + \sum_i c_i H(b_i) \right)$$

subject to:

$$\forall i, y_i, \alpha \in N(i), \sum_{y_\alpha \backslash y_i} b_\alpha(y_\alpha) = b_i(y_i)$$

LP Relaxation

- Replace the integrality constraint

$$\max \sum_{\alpha, y_\alpha} b_\alpha(y_\alpha)\theta_\alpha(y_\alpha) + \sum_{i, y_i} b_i(y_i)\theta_i(y_i)$$

subject to:

$$b_i(y_i), b_\alpha(y_\alpha) \in [0,1], \quad \sum_{y_\alpha} b_\alpha(y_\alpha) = 1, \sum_{y_i} b_i(y_i) = 1$$

$$\forall i, y_i, \alpha \in N(i), \sum_{y_\alpha \setminus y_i} b_\alpha(y_\alpha) = b_i(y_i)$$

- Introduce entropy barrier functions to be smooth and get rid of the simplex constraints

$$\max \sum_{\alpha, y_\alpha} b_\alpha(y_\alpha)\theta_\alpha(y_\alpha) + \sum_{i, y_i} b_i(y_i)\theta_i(y_i) + \epsilon\left(\sum_\alpha c_\alpha H(b_\alpha) + \sum_i c_i H(b_i)\right)$$

subject to:

$$\forall i, y_i, \alpha \in N(i), \sum_{y_\alpha \setminus y_i} b_\alpha(y_\alpha) = b_i(y_i)$$

Primal Formulation

- The dual problem

$$\max \sum_{\alpha, y_\alpha} b_\alpha(y_\alpha)\theta_\alpha(y_\alpha) + \sum_{i, y_i} b_i(y_i)\theta_i(y_i) + \epsilon \left(\sum_\alpha c_\alpha H(b_\alpha) + \sum_i c_i H(b_i) \right)$$

$$\text{subject to:} \qquad \forall i, y_i, \alpha \in N(i), \sum_{y_\alpha \setminus y_i} b_\alpha(y_\alpha) = b_i(y_i)$$

- Its primal problem has Lagrange multipliers for each constraint

$$\sum_\alpha \epsilon c_\alpha \ln \sum_{y_\alpha} \exp \left(\frac{\theta_\alpha(y_\alpha) + \sum_{i \in N(\alpha)} \lambda_{i \to \alpha}(y_i)}{\epsilon c_\alpha} \right) + \sum_i \epsilon c_i \ln \sum_{y_i} \exp \left(\frac{\theta_i(y_i) - \sum_{\alpha \in N(i)} \lambda_{i \to \alpha}(y_i)}{\epsilon c_i} \right)$$

Primal Formulation

- The dual problem

$$\max \sum_{\alpha, y_\alpha} b_\alpha(y_\alpha)\theta_\alpha(y_\alpha) + \sum_{i, y_i} b_i(y_i)\theta_i(y_i) + \epsilon \left(\sum_\alpha c_\alpha H(b_\alpha) + \sum_i c_i H(b_i) \right)$$

$$\text{subject to:} \qquad \forall i, y_i, \alpha \in N(i), \sum_{y_\alpha \setminus y_i} b_\alpha(y_\alpha) = b_i(y_i)$$

- Its primal problem has Lagrange multipliers for each constraint

$$\sum_\alpha \epsilon c_\alpha \ln \sum_{y_\alpha} \exp\left(\frac{\theta_\alpha(y_\alpha) + \sum_{i \in N(\alpha)} \lambda_{i \to \alpha}(y_i)}{\epsilon c_\alpha} \right) + \sum_i \epsilon c_i \ln \sum_{y_i} \exp\left(\frac{\theta_i(y_i) - \sum_{\alpha \in N(i)} \lambda_{i \to \alpha}(y_i)}{\epsilon c_i} \right)$$

- Optimization via coordinate descent (close form updates) gives us a new family of message passing algorithms [Hazan and Sashua 2010].
- Optimum guaranteed when $c_i, c_\alpha > 0$.

Primal Formulation

- The dual problem

$$\max \sum_{\alpha, y_\alpha} b_\alpha(y_\alpha)\theta_\alpha(y_\alpha) + \sum_{i, y_i} b_i(y_i)\theta_i(y_i) + \epsilon \left(\sum_\alpha c_\alpha H(b_\alpha) + \sum_i c_i H(b_i) \right)$$

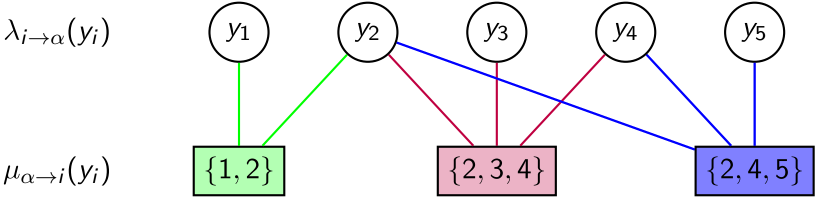
$$\text{subject to:} \qquad \forall i, y_i, \alpha \in N(i), \sum_{y_\alpha \setminus y_i} b_\alpha(y_\alpha) = b_i(y_i)$$

- Its primal problem has Lagrange multipliers for each constraint

$$\sum_\alpha \epsilon c_\alpha \ln \sum_{y_\alpha} \exp\left(\frac{\theta_\alpha(y_\alpha) + \sum_{i \in N(\alpha)} \lambda_{i \to \alpha}(y_i)}{\epsilon c_\alpha} \right) + \sum_i \epsilon c_i \ln \sum_{y_i} \exp\left(\frac{\theta_i(y_i) - \sum_{\alpha \in N(i)} \lambda_{i \to \alpha}(y_i)}{\epsilon c_i} \right)$$

- Optimization via coordinate descent (close form updates) gives us a new family of message passing algorithms [Hazan and Sashua 2010].
- Optimum guaranteed when $c_i, c_\alpha > 0$.

Message passing: convex max-product when $\epsilon = 0$



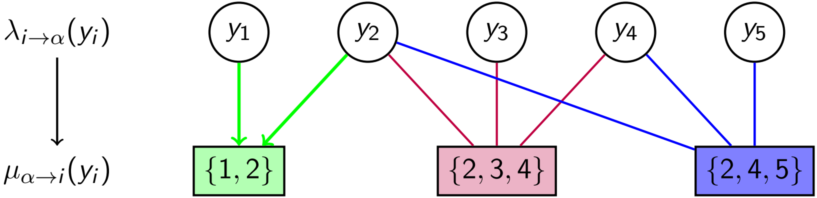
Algorithm 1 Convex Max-Product:

Set $\hat{c}_i = c_i + \sum_{\alpha \in N(i)} c_\alpha$. For every $i = 1, \ldots, n$ repeat until convergence: $\forall \alpha \in N(i), x_i$:

$$\mu_{\alpha\to i}(x_i) = \inf_{x_\alpha \setminus x_i} \left\{ \theta_\alpha(x_\alpha) + \sum_{j \in N(\alpha)\setminus i} \lambda_{j\to\alpha}(x_j) \right\}$$

$$\lambda_{i\to\alpha}(x_i) = \frac{c_\alpha}{\hat{c}_i} \left(\theta_i(x_i) + \sum_{\beta \in N(i)} \mu_{\beta\to i}(x_i) \right) - \mu_{\alpha\to i}(x_i)$$

Message passing: convex max-product when $\epsilon = 0$



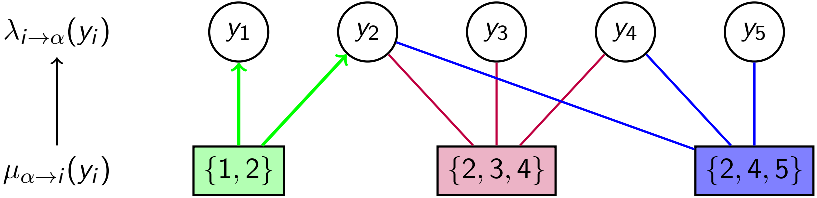
Algorithm 1 Convex Max-Product:

Set $\hat{c}_i = c_i + \sum_{\alpha \in N(i)} c_\alpha$. For every $i = 1, ..., n$ repeat until convergence: $\forall \alpha \in N(i), x_i$:

$$\mu_{\alpha \to i}(x_i) = \inf_{x_\alpha \setminus x_i} \left\{ \theta_\alpha(x_\alpha) + \sum_{j \in N(\alpha) \setminus i} \lambda_{j \to \alpha}(x_j) \right\}$$

$$\lambda_{i \to \alpha}(x_i) = \frac{c_\alpha}{\hat{c}_i} \left(\theta_i(x_i) + \sum_{\beta \in N(i)} \mu_{\beta \to i}(x_i) \right) - \mu_{\alpha \to i}(x_i)$$

Message passing: convex max-product when $\epsilon = 0$

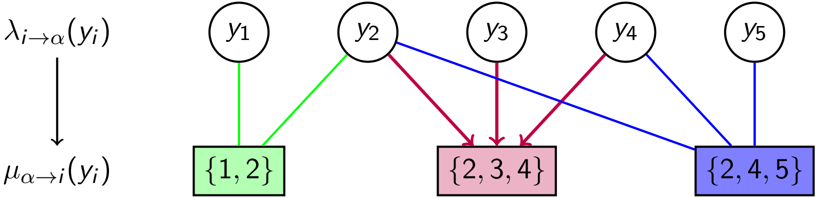


Algorithm 1 Convex Max-Product:

Set $\hat{c}_i = c_i + \sum_{\alpha \in N(i)} c_\alpha$. For every $i = 1, ..., n$ repeat until convergence: $\forall \alpha \in N(i), x_i$:

$$\mu_{\alpha\to i}(x_i) = \inf_{x_\alpha \setminus x_i} \left\{ \theta_\alpha(x_\alpha) + \sum_{j \in N(\alpha)\setminus i} \lambda_{j\to\alpha}(x_j) \right\}$$

$$\lambda_{i\to\alpha}(x_i) = \frac{c_\alpha}{\hat{c}_i} \left(\theta_i(x_i) + \sum_{\beta \in N(i)} \mu_{\beta\to i}(x_i) \right) - \mu_{\alpha\to i}(x_i)$$



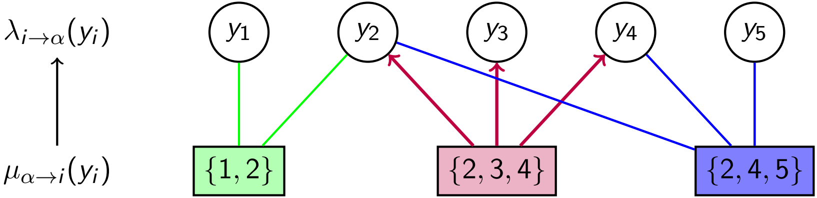
Algorithm 1 Convex Max-Product:

Set $\hat{c}_i = c_i + \sum_{\alpha \in N(i)} c_\alpha$. For every $i = 1, ..., n$ repeat until convergence: $\forall \alpha \in N(i), x_i$:

$$\mu_{\alpha\to i}(x_i) = \inf_{x_\alpha \setminus x_i} \left\{ \theta_\alpha(x_\alpha) + \sum_{j \in N(\alpha)\setminus i} \lambda_{j\to\alpha}(x_j) \right\}$$

$$\lambda_{i\to\alpha}(x_i) = \frac{c_\alpha}{\hat{c}_i} \left(\theta_i(x_i) + \sum_{\beta \in N(i)} \mu_{\beta\to i}(x_i) \right) - \mu_{\alpha\to i}(x_i)$$

Message passing: convex max-product when $\epsilon = 0$



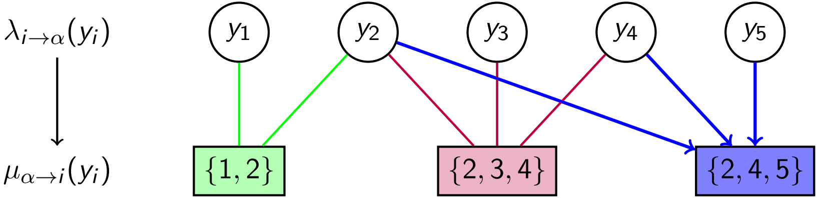
Algorithm 1 Convex Max-Product:

Set $\hat{c}_i = c_i + \sum_{\alpha \in N(i)} c_\alpha$. For every $i = 1, ..., n$ repeat until convergence: $\forall \alpha \in N(i), x_i$:

$$\mu_{\alpha\to i}(x_i) = \inf_{x_\alpha \backslash x_i} \left\{ \theta_\alpha(x_\alpha) + \sum_{j \in N(\alpha)\backslash i} \lambda_{j\to\alpha}(x_j) \right\}$$

$$\lambda_{i\to\alpha}(x_i) = \frac{c_\alpha}{\hat{c}_i} \left(\theta_i(x_i) + \sum_{\beta \in N(i)} \mu_{\beta\to i}(x_i) \right) - \mu_{\alpha\to i}(x_i)$$

Message passing: convex max-product when $\epsilon = 0$



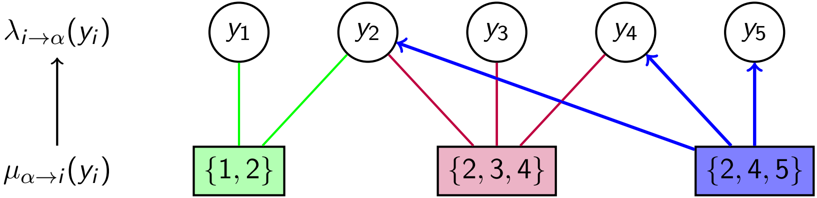
Algorithm 1 Convex Max-Product:

Set $\hat{c}_i = c_i + \sum_{\alpha \in N(i)} c_\alpha$. For every $i = 1, ..., n$ repeat until convergence: $\forall \alpha \in N(i), x_i$:

$$\mu_{\alpha \to i}(x_i) = \inf_{x_\alpha \setminus x_i} \left\{ \theta_\alpha(x_\alpha) + \sum_{j \in N(\alpha) \setminus i} \lambda_{j \to \alpha}(x_j) \right\}$$

$$\lambda_{i \to \alpha}(x_i) = \frac{c_\alpha}{\hat{c}_i} \left(\theta_i(x_i) + \sum_{\beta \in N(i)} \mu_{\beta \to i}(x_i) \right) - \mu_{\alpha \to i}(x_i)$$

Message passing: convex max-product when $\epsilon = 0$



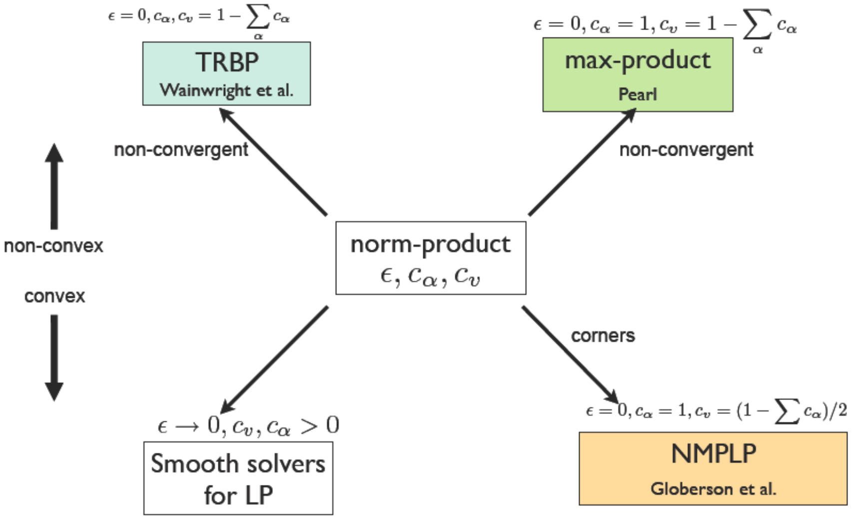
Algorithm 1 Convex Max-Product:

Set $\hat{c}_i = c_i + \sum_{\alpha \in N(i)} c_\alpha$. For every $i = 1, ..., n$ repeat until convergence: $\forall \alpha \in N(i), x_i$:

$$\mu_{\alpha\to i}(x_i) = \inf_{x_\alpha \setminus x_i} \left\{ \theta_\alpha(x_\alpha) + \sum_{j \in N(\alpha)\setminus i} \lambda_{j\to\alpha}(x_j) \right\}$$

$$\lambda_{i\to\alpha}(x_i) = \frac{c_\alpha}{\hat{c}_i} \left(\theta_i(x_i) + \sum_{\beta \in N(i)} \mu_{\beta\to i}(x_i) \right) - \mu_{\alpha\to i}(x_i)$$

Equivalence with other models



Cloud computing: Very large problems

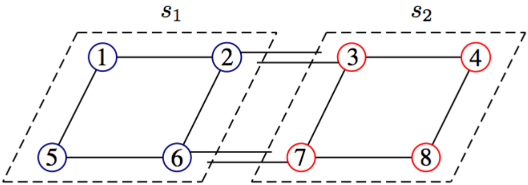
- Dual decomposition: partition the problem and add Lagrange multipliers

$$\max \sum_{s \in G_{\mathcal{P}}} \sum_{\alpha \in G_s, x_\alpha} b_\alpha^s(x_\alpha)\theta_\alpha(x_\alpha) + \sum_{i \in G_s, x_i} b_i^s(x_i)\theta_i(x_i) + \epsilon \sum_{s \in G_{\mathcal{P}}} \left(\sum_{\alpha \in G_s} c_\alpha H(b_\alpha^s) + \sum_{i \in G_s} c_i H(b_i^s) \right)$$

subject to:

$$\forall s, i, x_i, \alpha \in N(i), \quad \sum_{x_\alpha \setminus x_i} b_\alpha^s(x_\alpha) = b_i^s(x_i)$$

$$\forall s, \alpha \in N_{\mathcal{P}}(s), x_\alpha, \quad b_\alpha^s(x_\alpha) = b_\alpha(x_\alpha)$$



- We obtain a dual problem with one additional set of Lagrange multipliers which are messages between machines [Schwing et al. 11].