Probabilistic Graphical Models and Their Applications

Bjoern Andres and Bernt Schiele

Max Planck Institute for Informatics

slides adapted from Peter Gehler

November 23, 2016
Today’s topics

- Inference
  - Sum-Product (brief recap)
  - Max-Product
- Application
  - Human Pose Estimation
Relationship Potentials to Graphs

- **Factor Graphs:**
  - (a) (b) (c)

- Left: Markov Network
- Middle: Factor graph representation of $\phi(a, b, c)$
- Right: Factor graph representation of $\phi(a, b)\phi(b, c)\phi(c, a)$
- Different factor graphs can have the same Markov network $(b,c) \Rightarrow (a)$
Inference in Trees
Inference - what to infer?

- Given distribution
  \[ p(x) = p(x_1, \ldots, x_n) \]  
  \[ (1) \]

- Inference: computing functions of the distribution, e.g.
  - mean
  - mode
  - marginal
  - conditionals
What to infer?

- **Mean**
  \[
  \mathbb{E}_{p(x)}[x] = \sum_{x \in \mathcal{X}} xp(x)
  \]

- **Mode (most likely state)**
  \[
  x^* = \arg\max_{x \in \mathcal{X}} p(x)
  \]

- **Conditional Distributions**
  \[
  p(x_i, x_j \mid x_k, x_l) \quad \text{or} \quad p(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)
  \]

- **Max-Marginals**
  \[
  x^*_i = \arg\max_{x_i \in \mathcal{X}_i} p(x_i) = \arg\max_{x_i \in \mathcal{X}_i} \sum_{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n} p(x)
  \]
Example: Pictorial Structures for Human Pose Estimation

- Find body parts (i.e. find 2D locations and orientations of head, torso, lower/upper left/right arms/legs)

- Inference for human body pose estimation:
  - Calculating marginals (sum-product algorithm):
    \[
    \arg\max_{x_i \in \mathcal{X}_i} p(x_i) = \arg\max_{x_i \in \mathcal{X}_i} \sum_{x_i \in \mathcal{X}_i} (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) p(x)
    \]
  - Calculating mode (max-product algorithm):
    \[
    x^* = \arg\max_{x \in \mathcal{X}} p(x)
    \]
Consider a branching graph:

with factors

$$f_1(a, b) f_2(b, c, d) f_3(c) f_4(d, e) f_5(d)$$  \(2\)

For example: find marginal \(p(a)\)
General singly-connected factor graphs – 2

- Idea: compute messages
Sum-Product Algorithm – Overview

- Algorithm to compute all messages efficiently
- Assuming the graph is singly-connected

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

- Then compute any desired marginals
- Also known as belief propagation
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)

\[ \mu_{f \rightarrow x}(x) = f(x) \quad \mu_{x \rightarrow f}(x) = 1 \]
2. Variable to Factor message

\[
\mu_{x \rightarrow f}(x) = \prod_{g \in \{\text{ne}(x) \setminus f\}} \mu_{g \rightarrow x}(x)
\]  

(3)
3. Factor to Variable message

\[ \mu_{f \rightarrow x}(x) = \sum_{y \in \mathcal{X}_f \setminus x} \phi_f(\mathcal{X}_f) \prod_{y \in \{ \text{ne}(f) \setminus x \}} \mu_{y \rightarrow f}(y) \quad (4) \]

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

\[ p(x) \propto \prod_{f \in \text{ne}(x)} \mu_{f \rightarrow x}(x) \]
So far..

- So far marginals
- Now: finding the maximal state (mode)
Finding the maximal state: Max-Product

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

$$x^* = \arg\max_{x_1, \ldots, x_n} p(x_1, \ldots, x_n)$$  \hspace{1cm} (6)

- Again use factorization structure to distribute the maximisation to local computations

- Example: chain

$$f(a, b, c, d) = \phi(a, b)\phi(b, c)\phi(c, d)$$  \hspace{1cm} (7)
Be careful: not maximal marginal states!

- The most likely state

\[ x^* = \arg\max_{x_1, \ldots, x_n} p(x_1, \ldots, x_n) \]  

(8)

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

- For all \( i = 1, \ldots, n \)

\[ x_i^* = \arg\max_{x_i} p(x_i) \]  

(9)

Example:

<table>
<thead>
<tr>
<th></th>
<th>( x_1 = 0 )</th>
<th>( x_1 = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_2 = 0 )</td>
<td>0.3</td>
<td>0.4</td>
</tr>
<tr>
<td>( x_2 = 1 )</td>
<td>0.3</td>
<td>0.0</td>
</tr>
<tr>
<td>marginal ( p(x_1) )</td>
<td>0.6</td>
<td>0.4</td>
</tr>
</tbody>
</table>
Example: Chain

\[
\begin{align*}
\max_x f(x) & = \max_{x_1, x_2, x_3, x_4} \phi(x_1, x_2)\phi(x_2, x_3)\phi(x_3, x_4) \\
& = \max_{x_1, x_2, x_3} \phi(x_1, x_2)\phi(x_2, x_3) \max_{x_4} \phi(x_3, x_4) \\
& = \max_{x_1, x_2} \phi(x_1, x_2) \max_{x_3} \phi(x_2, x_3) \gamma(x_3) \\
& = \max_{x_1} \max_{x_2} \phi(x_1, x_2) \gamma(x_2) \\
& = \max_{x_1} \gamma(x_1)
\end{align*}
\]
Example: Chain

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

\[
\begin{align*}
x_1^* &= \argmax_{x_1} \gamma(x_1) \\
x_2^* &= \argmax_{x_2} \phi(x_1^*, x_2) \gamma(x_2) \\
x_3^* &= \argmax_{x_3} \phi(x_2^*, x_3) \gamma(x_3) \\
x_4^* &= \argmax_{x_4} \phi(x_3^*, x_4) \gamma(x_4)
\end{align*}
\]

- this is called backtracking (an application of dynamic programming)
Spot the messages:

\[
\begin{align*}
\max_x f(x) &= \max_{a,b,c,d,e} f_1(a,b) f_2(b,c,d) f_3(c) f_4(d,e) f_5(d) \\
&= ?
\end{align*}
\]
\[
\max_x f(x) = \max_{a,b,c,d,e} f_1(a,b) f_2(b,c,d) f_3(c) f_4(d,e) f_5(d) \\
= \max_a \max_b f_1(a,b) \max_{c,d} f_2(b,c,d) f_3(c) \underbrace{f_5(d)}_{\mu_{f_5 \to d}(d)} \max_e f_4(d,e) \\
= \max_a \max_b f_1(a,b) \max_{c,d} f_2(b,c,d) \underbrace{f_3(c)}_{\mu_{c \to f_2}(c)} \underbrace{f_5(d)}_{\mu_{f_5 \to d}(d)} \underbrace{f_4(d,e)}_{\mu_{f_4 \to d}(d)} \\
= \max_a \max_b f_1(a,b) \max_{c,d} f_2(b,c,d) \mu_{c \to f_2}(c) \mu_{d \to f_2}(d) \underbrace{f_5(d)}_{\mu_{f_5 \to b}(b)}
\]
max \, 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 \max_b \, f_1(a,b) \, \mu_{f_2\rightarrow b}(b) \\
= \max_a \max_b \, f_1(a,b) \, \mu_{b\rightarrow f_1}(b) \\
= \max_a \mu_{f_1\rightarrow a}(a)
Max-Product Algorithm

- So we need an algorithm to compute the messages
- Pick any variable as root

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
1. Initialisation

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

\[ \mu_{f \rightarrow x}(x) = f(x) \]

\[ \mu_{x \rightarrow f}(x) = 1 \]

- Same as for sum-product
2. Variable to Factor message

\[ \mu_{x \rightarrow f}(x) = \prod_{g \in \{ \text{ne}(x) \setminus f \}} \mu_{g \rightarrow x}(x) \]  

\[ \mu_{f_1 \rightarrow x}(x) \]
\[ \mu_{f_2 \rightarrow x}(x) \]
\[ \mu_{f_3 \rightarrow x}(x) \]

- Same as for sum-product

Inference in Trees
3. Factor to Variable message

\[
\mu_{f \rightarrow x}(x) = \max_{y \in \mathcal{X}_f \setminus x} \phi_f(\mathcal{X}_f) \prod_{y \in \text{ne}(f) \setminus x} \mu_{y \rightarrow f}(y) \tag{11}
\]

- Different message than in sum-product
- This is now a max-product
Maximal state of Variable

\[ x^* = \arg\max_x \prod_{f \in \text{ne}(x)} \mu_{f \rightarrow x}(x) \]
Comments

- Products of small probabilities may lead to numerical instabilities
- Take the logarithm

\[
\ln \left( \max_x p(x) \right) = \max_x \ln p(x)
\]  

(13)

- Taking the logarithm replaces the products with sums (yields the max-sum algorithm)
Example: Pictorial Structures

- An Application: Human Body Pose Estimation
  - Find Locations of Body Parts

[Fischler & Elschlager, 1973], [Felsenzwalb & Huttenlocher, 2000]
Inference in Trees

Pictorial Structures

- Each body part one variable (torso, head, etc) (11 total)
- Each variable represented as tuple e.g. 
  \( y_{torso} = (x, y, s, \theta) \)
  - \((x, y)\) image coordinates
  - \(s\) scale
  - \(\theta\) rotation of the part
- Discretize label space \( y \) (that is \( x, y, s, \theta \)) in \( L \) states
  - size of \( L \) e.g. 
    \[ L \approx 500,000 = 125 \times 125 \times 4 \times 8 = x_{pos.} \times y_{pos.} \times scales \times orientations \]

[Felsenzwalb & Huttenlocher, 2000]
Pictorial Structures

- **Potentials:**
  \[ E_{\text{torso}}(y_{\text{torso}}, X), E_{\text{rarm}}(y_{\text{rarm}}, X), \ldots, \]

- **Pairwise potentials:**
  \[ E_{\text{torso,rarm}}(y_{\text{torso}}, y_{\text{rarm}}), \ldots, \]

- \( k \) be the number of parts (11), \( L \) the size of the label space (\( \approx 500,000 \))

- Given new test image, max-product algorithm complexity is \( \mathcal{O}(kL^2) \)

- For specific potentials reduction to \( \mathcal{O}(kL) \)

[Felsenzwalb& Huttenlocher, 2000]