Summary from last week
Data Integration

Display Advertising

Webpage Ranking

Answers

News

Wednesday, January 20, 2010
Message Passing

- Forward/Backward messages as normal for chain
- When we have more edges for a vertex use
  - For each outgoing message, send it once you have all other incoming messages
- PRINCIPLED HACK
  If no message received yet, set it to 1 altogether
Message Passing

- Forward/Backward messages as normal for chain
- When we have more edges for a vertex use
  - For each outgoing message, send it once you have all other incoming messages
- **PRINCIPLED HACK**
  If no message received yet, set it to 1 altogether
Message Passing

- Forward/Backward messages as normal for chain
- When we have more edges for a vertex use
  - For each outgoing message, send it once you have all other incoming messages
- PRINCIPLEd HACK
  If no message received yet, set it to 1 altogether
Message Passing

- Forward/Backward messages as normal for chain
- When we have more edges for a vertex use
  - For each outgoing message, send it once you have all other incoming messages
- **PRINCIPLED HACK**
  - If no message received yet, set it to 1 altogether
Message Passing

• Forward/Backward messages as normal for chain
• When we have more edges for a vertex use
  • For each outgoing message, send it once you have all other incoming messages
• PRINCIPLED HACK
  If no message received yet, set it to 1 altogether
Message Passing

- Forward/Backward messages as normal for chain
- When we have more edges for a vertex use
  - For each outgoing message, send it once you have all other incoming messages
- **PRINCIPLED HACK**
  If no message received yet, set it to 1 altogether

\[
\begin{align*}
X_0 & \rightarrow X_1 & \rightarrow X_2 & \rightarrow X_3 & \rightarrow X_4 & \rightarrow X_5 \\
& \rightarrow X_6 & \rightarrow X_7 & \rightarrow X_8
\end{align*}
\]
Message Passing

- Forward/Backward messages as normal for chain
- When we have more edges for a vertex use
  - For each outgoing message, send it once you have all other incoming messages
- **PRINCIPLED HACK**
  If no message received yet, set it to 1 altogether
Losing the direction ...
Chicken and Egg
Chicken and Egg

\[ p(c|e)p(e|c) \]
Chicken and Egg

\[ p(c|e)p(e|c) \]
we know that chicken and egg are correlated

chicken and egg
no chicken and no egg

$p(c, e) \propto \exp \psi(c, e)$

encode the correlation via the clique potential between c and e
we know that chicken and egg are correlated

either chicken or egg

\[ p(c, e) = \frac{\exp \psi(c, e)}{\sum_{c', e'} \exp \psi(c', e')} \]

\[ = \exp [\psi(c, e) - g(\psi)] \quad \text{where} \quad g(\psi) = \log \sum_{c, e} \exp \psi(c, e) \]
... some Yahoo service
... some Yahoo service

MySQL

Apache

Website

\[ p(w|m, a)p(m)p(a) \]
... some Yahoo service

MySQL -> Website

Apache -> Website

\[ p(w|m, a)p(m)p(a) \]

\[ m \not\perp a|w \]
... some Yahoo service

\[ p(w|m, a)p(m)p(a) \]
\[ m \not\perp a | w \]

\[ p(m, w, a) \propto \phi(m, w)\phi(w, a) \]
... some Yahoo service

\[ p(w|m, a)p(m)p(a) \]
\[ m \not\perp a | w \]

\[ p(m, w, a) \propto \phi(m, w)\phi(w, a) \]
... some Yahoo service

\[
p(w|m, a)p(m)p(a)
\]
\[
m \not\perp a|w
\]

\[
p(m, w, a) \propto \phi(m, w)\phi(w, a)
\]
\[
m \perp a|w
\]
... some Yahoo service

MySQL  →  Website  →  Apache

$p(w|m, a)p(m)p(a)$
$m ⊥ a | w$

easier "debugging"

MySQL  →  Website

$\text{Site affects}
\text{MySQL}$

$p(m, w, a) \propto \phi(m, w)\phi(w, a)$
$m ⊥⊥ a | w$

easier "modeling"
Key Concept
Observing nodes makes remainder conditionally independent
Undirected Graphical Models

Key Concept
Observing nodes makes remainder conditionally independent
Key Concept
Observing nodes makes remainder conditionally independent
Undirected Graphical Models

Key Concept
Observing nodes makes remainder conditionally independent
Undirected Graphical Models

Key Concept
Observing nodes makes remainder conditionally independent
Undirected Graphical Models

Key Concept
Observing nodes makes remainder conditionally independent
Undirected Graphical Models

Key Concept
Observing nodes makes remainder conditionally independent
Key Concept
Observing nodes makes remainder conditionally independent
Cliques

maximal fully connected subgraph
Cliques

maximal fully connected subgraph
Cliques

maximal fully connected subgraph
Hammersley Clifford Theorem

If density has full support then it decomposes into products of clique potentials

\[ p(x) = \prod_c \psi_c(x_c) \]
Directed vs. Undirected

- Causal description
- Normalization automatic
- Intuitive
- Requires knowledge of dependencies
- Conditional independence tricky (Bayes Ball algorithm)

- Noncausal description (correlation only)
- Intuitive
- Easy modeling
- Normalization difficult
- Conditional independence easy to read off (graph connectivity)
Examples
Chains

\[ p(x) = \prod_i \psi_i(x_i, x_{i+1}) \]
Chains

\[
p(x) = \prod_i \psi_i(x_i, x_{i+1})
\]
Chains

\[ p(x) = \prod_{i} \psi_i(x_i, x_{i+1}) \]

\[ p(x, y) = \prod_{i} \psi_i^x(x_i, x_{i+1})\psi_i^{xy}(x_i, y_i) \]
Chains

\[ p(x) = \prod_i \psi_i(x_i, x_{i+1}) \]

\[ p(x|y) \propto \prod_i \left[ \psi^x_i(x_i, x_{i+1}) \psi^{xy}_i(x_i, y_i) \right] = f_i(x_i, x_{i+1}) \]

\[ p(x, y) = \prod_i \psi^x_i(x_i, x_{i+1}) \psi^{xy}_i(x_i, y_i) \]
Chains

\[ p(x|y) \propto \prod_i \psi_i^x(x_i, x_{i+1})\psi_i^{xy}(x_i, y_i) =: f_i(x_i, x_{i+1}) \]

Dynamic Programming

\[ l_1(x_1) = 1 \text{ and } l_{i+1}(x_{i+1}) = \sum_{x_i} l_i(x_i) f_i(x_i, x_{i+1}) \]

\[ r_n(x_n) = 1 \text{ and } r_i(x_i) = \sum_{x_{i+1}} r_{i+1}(x_{i+1}) f_i(x_i, x_{i+1}) \]
Named Entity Tagging

\[ p(x|y) \propto \prod_i \psi_i^x(x_i, x_{i+1})\psi_i^{xy}(x_i, y_i) =: f_i(x_i, x_{i+1}) \]

Wednesday, January 20, 2010

Diagram showing a network structure with nodes and edges representing entities and relationships.
• Ontology classification (e.g. YDir, DMOZ)

\[ p(y|\mathbf{x}) = \prod_{i} \psi(y_i, y_{\text{parent}(i)}, \mathbf{x}) \]
Spin Glasses + Images

\[ p(x|y) = \prod_{ij} \psi_{\text{right}}(x_{ij}, x_{i+1,j}) \psi_{\text{up}}(x_{ij}, x_{i,j+1}) \psi_{xy}(x_{ij}, y_{ij}) \]
Spin Glasses + Images

\[ p(x | y) = \prod_{ij} \psi_{\text{right}}(x_{ij}, x_{i+1,j}) \psi_{\text{up}}(x_{ij}, x_{i,j+1}) \psi_{xy}(x_{ij}, y_{ij}) \]
Image Denoising

Li&Huttenlocher, ECCV’08
Semi-Markov Models

- Flexible length of an episode
- Segmentation between episodes

phrase segmentation, activity recognition, motion data analysis
Shi, Smola, Altun, Vishwanathan, Li, 2007-2009
2D CRF for Webpages

web page information extraction, segmentation, annotation
Bo, Zhu, Nie, Wen, Hon, 2005-2007
• **Density function**

\[ p(x; \theta) = \exp (\langle \phi(x), \theta \rangle - g(\theta)) \]

where \( g(\theta) = \log \sum_{x'} \exp (\langle \phi(x'), \theta \rangle) \)

• **Log partition function generates cumulants**

\[ \partial_\theta g(\theta) = \mathbf{E} [\phi(x)] \]

\[ \partial_\theta^2 g(\theta) = \text{Var} [\phi(x)] \]

• \( g \) is convex (second derivative is p.s.d.)
Log Partition Function

Unconditional model

\[ p(x|\theta) = e^{\langle \phi(x), \theta \rangle} - g(\theta) \]
\[ g(\theta) = \log \sum_x e^{\langle \phi(x), \theta \rangle} \]
\[ \partial_\theta g(\theta) = \frac{\sum_x \phi(x)e^{\langle \phi(x), \theta \rangle}}{\sum_x e^{\langle \phi(x), \theta \rangle}} = \sum_x \phi(x)e^{\langle \phi(x), \theta \rangle} - g(\theta) \]

Conditional model

\[ p(y|x, \theta) = e^{\langle \phi(x,y), \theta \rangle} - g(\theta|x) \]
\[ g(\theta|x) = \log \sum_y e^{\langle \phi(x,y), \theta \rangle} \]
\[ \partial_\theta g(\theta|x) = \frac{\sum_y \phi(x,y)e^{\langle \phi(x,y), \theta \rangle}}{\sum_y e^{\langle \phi(x,y), \theta \rangle}} = \sum_y \phi(x,y)e^{\langle \phi(x,y), \theta \rangle} - g(\theta|x) \]
Estimation

• Conditional log-likelihood

\[
\log p(y|x; \theta) = \langle \phi(x, y), \theta \rangle - g(\theta|x)
\]

• Log-posterior (Gaussian Prior)

\[
\log p(\theta|X, Y) = \sum_i \log(y_i|x_i; \theta) + \log p(\theta) + \text{const.}
\]

\[
= \left\langle \sum_i \phi(x_i, y_i), \theta \right\rangle - \sum_i g(\theta|x_i) - \frac{1}{2\sigma^2} ||\theta||^2 + \text{const.}
\]

• First order optimality conditions

\[
\sum_i \phi(x_i, y_i) = \sum_i \mathbb{E}_{y|x_i} [\phi(x_i, y)] + \frac{1}{\sigma^2} \theta
\]
Logistic Regression

• Label space

\[ \phi(x, y) = y\phi(x) \text{ where } y \in \{\pm 1\} \]

• Log-partition function

\[ g(\theta|x) = \log \left[ e^{\langle \phi(x), \theta \rangle} + e^{-\langle \phi(x), \theta \rangle} \right] = \log 2 \cosh \langle \phi(x), \theta \rangle \]

• Convex minimization problem

\[ \text{minimize } \frac{1}{2\sigma^2} \left\| \theta \right\|^2 + \sum_i \log 2 \cosh \langle \phi(x_i), \theta \rangle - y_i \langle \phi(x_i), \theta \rangle \]

• Prediction

\[ p(y|x, \theta) = \frac{e^{y\langle \phi(x), \theta \rangle}}{e^{\langle \phi(x), \theta \rangle} + e^{-\langle \phi(x), \theta \rangle}} = \frac{1}{1 + e^{-2y\langle \phi(x), \theta \rangle}} \]
Exponential Clique Decomposition

$p(x) = \prod_c \psi_c(x_c)$

**Theorem:** Clique decomposition holds in sufficient statistics

$\phi(x) = (\ldots, \phi_c(x_c), \ldots)$ and $\langle \phi(x), \theta \rangle = \sum_c \langle \phi_c(x_c), \theta_c \rangle$

**Corollary:** we only need expectations on cliques

$E_x[\phi(x)] = (\ldots, E_{x_c}[\phi_c(x_c)], \ldots)$
Conditional Random Fields

\[ \phi(x) = (y_1 \phi_x(x_1), \ldots, y_n \phi_x(x_n), \phi_y(y_1, y_2), \ldots, \phi_y(y_{n-1}, y_n)) \]

\[ \langle \phi(x), \theta \rangle = \sum_i \langle \phi_x(x_i, y_i), \theta_x \rangle + \sum_i \langle \phi_y(y_i, y_{i+1}), \theta_y \rangle \]

\[ g(\theta|x) = \sum_y \prod_i f_i(y_i, y_{i+1}) \text{ where} \]

\[ f_i(y_i, y_{i+1}) = e^{\langle \phi_x(x_i, y_i), \theta_x \rangle + \langle \phi_y(y_i, y_{i+1}), \theta_y \rangle} \]
Conditional Random Fields

• Compute distribution over marginal and adjacent labels
• Take conditional expectations
• Take update step (batch or online)

• More general techniques for computing normalization via message passing ...
Dynamic Programming + Message Passing
Clique Graph

\[ p(x) = \prod_c \psi_c(x_c) \]
Clique Graph

\[ p(x) = \prod_c \psi_c(x_c) \]
Clique Graph

\[ p(x) = \prod_c \psi_c(x_c) \]
Clique Graph

\[ p(x) = \prod_c \psi_c(x_c) \]
$p(x) = \prod_c \psi_c(x_c)$
Junction Tree / Triangulation

$$p(x) = \prod_c \psi_c(x_c)$$
Junction Tree / Triangulation

\[ p(x) = \prod_c \psi_c(x_c) \]

message passing possible
Triangulation Examples

- Clique size increases
- Separator set size increases
Triangulation Examples

- Clique size increases
- Separator set size increases
Triangulation Examples

- Clique size increases
- Separator set size increases
• Joint Probability

\[ p(x) \propto \psi(x_1, x_2, x_3)\psi(x_1, x_3, x_4)\psi(x_1, x_4, x_5)\psi(x_1, x_5, x_6)\psi(x_1, x_6, x_7) \]

• Computing the normalization

\[ m\rightarrow(x_1, x_3) = \sum_{x_2} \psi(x_1, x_2, x_3) \]

\[ m\rightarrow(x_1, x_4) = \sum_{x_3} m\rightarrow(x_1, x_3)\psi(x_1, x_3, x_4) \]

\[ m\rightarrow(x_1, x_5) = \sum_{x_4} m\rightarrow(x_1, x_4)\psi(x_1, x_4, x_5) \]
Message Passing

- Joint Probability
  
  \[ p(x) \propto \psi(x_1, x_2, x_3)\psi(x_1, x_3, x_4)\psi(x_1, x_4, x_5)\psi(x_1, x_5, x_6)\psi(x_1, x_6, x_7) \]

- Computing the normalization

  \[ m_{\rightarrow}(x_1, x_3) = \sum_{x_2} \psi(x_1, x_2, x_3) \]

  \[ m_{\rightarrow}(x_1, x_4) = \sum_{x_3} m_{\rightarrow}(x_1, x_3)\psi(x_1, x_3, x_4) \]

  \[ m_{\rightarrow}(x_1, x_5) = \sum_{x_4} m_{\rightarrow}(x_1, x_4)\psi(x_1, x_4, x_5) \]
Message Passing

- Initialize messages with 1
- Guaranteed to converge for (junction) trees
- Works well in practice even for loopy graphs
- Only local computations are required

\[
m_{c\to c'}(x_{c \cap c'}) = \sum_{x_{c \setminus c'}} \psi_c(x_c) \prod_{c'' \in N(c) \setminus c'} m_{c'' \to c}(x_{c \cap c''})
\]

\[
p(x_c) \propto \psi_c(x_c) \prod_{c'' \in N(c)} m_{c'' \to c}(x_{c \cap c''})
\]
Message Passing in Practice

• Incoming messages contain aggregate uncertainty from neighboring random variables
• Message passing combines and transmits this information in both directions

crawler tiering

crawler

phase 1 ranker

tiering
Message Passing in Practice

• Incoming messages contain aggregate uncertainty from neighboring random variables
• Message passing combines and transmits this information in both directions
Summary
undirected graphical models

- Hammersley Clifford Theorem
- Junction Trees
- Message Passing
- Conditional Random Fields
- Connections to Classification