CSE 586, Spring 2009
Advanced Computer Vision
Introduction to Graphical Models

Review: Probability Theory

- Sum rule (marginal distributions)
  \[ p(x) = \sum_y p(x, y) \]
- Product rule
  \[ p(x, y) = p(x|y)p(y) \]
- From these we have Bayes' theorem
  \[ p(y|x) = \frac{p(x|y)p(y)}{p(x)} \]
  - with normalization factor
  \[ p(x) = \sum_y p(x|y)p(y) \]

Review: Probability Theory

- Conditional Probability (rewriting product rule)
  \[ P(A | B) = \frac{P(A, B)}{P(B)} \]
- Chain Rule
  \[ P(A, B, C, D) = \frac{P(A)P(A, B)P(A, B, C)P(A, B, C, D)}{P(A)P(B | A)P(C | A, B)P(D | A, B, C)} \]
- Conditional Independence
  \[ P(A, B | C) = P(A | C)P(B | C) \]
  - statistical independence
  \[ P(A, B) = P(A)P(B) \]

Summary of Graphical Models

- Joint probability / factoring the joint
- Graphical Models model dependence/independence
- Directed graphs
  \[ p(x_1, \ldots, x_D) = \prod_{i=1}^{D} p(x_i | \pi_a_i) \]  
  Example: HMM
- Undirected graphs
  \[ p(x) = \frac{1}{Z} \prod_{C} \psi_C(x_C) \]  
  Example: MRF
- Inference by message passing: belief propagation
  - Sum-product algorithm
  - Max-product (Min-sum if using logs)

The Joint Distribution

Recipe for making a joint distribution of M variables:

Example: Boolean variables A, B, C
Recipe for making a joint distribution of M variables:

1. Make a truth table listing all combinations of values of your variables (if there are M Boolean variables then the table will have $2^M$ rows).

Example: Boolean variables A, B, C

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Prob</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.30</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.05</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0.10</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.05</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.05</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0.10</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.25</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.10</td>
</tr>
</tbody>
</table>

3. If you subscribe to the axioms of probability, those numbers must sum to 1.

Using the Joint

Once you have the JD you can ask for the probability of any logical expression involving your attribute

$$P(E) = \sum \text{P(row) for rows matching } E$$

$$P(\text{Poor Male}) = 0.4654$$

$$P(\text{Poor}) = 0.7604$$
Joint distributions

- **Good news**
  Once you have a joint distribution, you can answer all sorts of probabilistic questions involving combinations of attributes

- **Bad news**
  Impossible to create JD for more than about ten attributes because there are so many numbers needed when you build the thing.

  For 10 binary variables you need to specify $2^{10} - 1$ numbers = 1023.
  (question for class: why the -1?)

How to use Fewer Numbers

- Factor the joint distribution into a product of distributions over subsets of variables
- Identify (or just assume) independence between some subsets of variables
- Use that independence to simplify some of the distributions
- Graphical models provide a principled way of doing this.

Factoring

- Consider an arbitrary joint distribution
  \[ p(x, y, z) \]

- We can always factor it, by application of the chain rule
  \[ p(x, y, z) = p(x)p(y | z, x) = p(x)p(y | x)p(z | x, y) \]

  what this factored form looks like as a graphical model

Directed versus Undirected Graphs

**Directed Graph Examples:**
- Bayes nets
- HMMs

**Undirected Graph Examples:**
- MRFS

Note: The word “graphical” denotes the graph structure underlying the model, not the fact that you can draw a pretty picture of it (although that helps).
Graphical Model Concepts

- Nodes represent random variables.
- Edges represent conditional (in)dependence.
- Each node is annotated with a table of conditional probabilities wrt parents.

Directed Acyclic Graphs

- For today we only talk about directed acyclic graphs (can't follow arrows around in a cycle).
- We can "read" the factored form of the joint distribution immediately from a directed graph

\[ p(x_1, \ldots, x_D) = \prod_{i=1}^{D} p(x_i | \text{pa}_i) \]

where \( \text{pa}_i \) denotes the parents of \( i \)

Factoring Examples

- Joint distribution

\[ p(x_1, \ldots, x_D) = \prod_{i=1}^{D} p(x_i | \text{pa}_i) \]

where \( \text{pa}_i \) denotes the parents of \( i \)

- We can "read" the factored form of the joint distribution directly from a directed graph

\[ p(x_1, \ldots, x_D) = \prod_{i=1}^{D} p(x_i | \text{pa}_i) \]

where \( \text{pa}_i \) denotes the parents of \( i \)

- We can "read" the factored form of the joint distribution directly from the directed graph

\[ p(x_1, \ldots, x_D) = \prod_{i=1}^{D} p(x_i | \text{pa}_i) \]

where \( \text{pa}_i \) denotes the parents of \( i \)

Note: \( P(L,S,M) = P(L|S,M)P(S)P(M) \)

Factoring Examples

- Joint distribution

\[ p(x_1, \ldots, x_D) = \prod_{i=1}^{D} p(x_i | \text{pa}_i) \]

where \( \text{pa}_i \) denotes the parents of \( i \)

- We can "read" the factored form of the joint distribution directly from the directed graph

\[ p(x_1, \ldots, x_D) = \prod_{i=1}^{D} p(x_i | \text{pa}_i) \]

where \( \text{pa}_i \) denotes the parents of \( i \)

Note: \( P(L,R,M) = P(M)P(L | M)P(R | M) \)
Factoring Examples

- How many probabilities do we have to specify/learn (assuming each \( x_i \) is a binary variable)?

  - if fully connected, we would need \( 2^7-1 = 127 \)
  - but, for this connectivity, we need \( 1+1+8+4+2+4 = 21 \)

Note: If all nodes were independent, we would only need 7!

Time Series

Consider modeling a time series of sequential data \( x_1, x_2, \ldots, x_N \)

These could represent:
- locations of a tracked object over time
- observations of the weather each day
- spectral coefficients of a speech signal
- joint angles during human motion

Modeling Time Series

Simplest model of a time series is that all observations are independent.

\[
\begin{align*}
\Pr(x_1, x_2, x_3, x_4, \ldots) &= \Pr(x_1)\Pr(x_2|x_1)\Pr(x_3|x_2)\Pr(x_4|x_3)\ldots \\
\Pr(x_{n+1}|x_n, x_{n-1}) &= \Pr(x_{n+1} | x_n)
\end{align*}
\]

This would be appropriate for modeling successive tosses of a biased coin.

However, it doesn’t really treat the series as a sequence. That is, we could permute the ordering of the observations and not change a thing.

Modeling Time Series

In the most general case, we could use chain rule to say that any node is dependent on all previous nodes...

\[
\begin{align*}
\Pr(x_1, x_2, x_3, x_4, \ldots) &= \Pr(x_1)\Pr(x_2|x_1)\Pr(x_3|x_2)\Pr(x_4|x_3)\ldots \\
\Pr(x_{n+1}|x_n, x_{n-1}) &= \Pr(x_{n+1} | x_n, x_{n-1})
\end{align*}
\]

Look for an intermediate model between these two extremes.

Modeling Time Series

Markov assumption:
\[
\Pr(x_n | x_1, x_2, \ldots, x_{n-1}) = \Pr(x_n | x_{n-1})
\]

that is, assume all conditional distributions depend only on the most recent previous observation.

The result is a first-order Markov Chain

\[
\begin{align*}
\Pr(x_1, x_2, x_3, x_4, \ldots) &= \Pr(x_1)\Pr(x_2|x_1)\Pr(x_3|x_2)\Pr(x_4|x_3)\ldots \\
\Pr(x_{n+1}|x_n) &= \Pr(x_{n+1} | x_n)
\end{align*}
\]

Generalization: State-Space Models

You have a Markov chain of latent (unobserved) states. Each state generates an observation

Goal: Given a sequence of observations, predict the sequence of unobserved states that maximizes the joint probability.
Modeling Time Series

Examples of State Space models

- Hidden Markov model
- Kalman filter

\[
\begin{align*}
&x_1, x_2, x_3, x_4, \ldots, x_n, x_{n+1} \\
&y_1, y_2, y_3, y_4, \ldots, y_n, y_{n+1}
\end{align*}
\]

Message Passing

Message Passing: Belief Propagation

- Example: 1D chain
- Find marginal for a particular node
- for M-state nodes, cost is \(O(M^L)\)
- exponential in length of chain
- but, we can exploit the graphical structure (conditional independences)

Applicable to both directed and undirected graphs.

Key Idea of Message Passing

multiplication distributes over addition

\(a \cdot (b + c) = a \cdot b + a \cdot c\)

as a consequence:

\[
\sum_i \sum_j a_i b_j c_k = \sum_i \sum_j a_i b_j \left( \sum_k c_k \right)
\]

\[
= \sum_i a_i \left[ \sum_j b_j \left( \sum_k c_k \right) \right]
\]

Example

\[
\sum_i \sum_j \sum_k a_i b_j c_k =
\]

\[
\sum_i \sum_j \sum_k \left[ a_i b_j \sum_k c_k \right] =
\]

48 multiplications + 23 additions

5 multiplications + 6 additions

For message passing, this principle is applied to functions of random variables, rather than the variables as done here.
In the next several slides, we will consider an example of a simple, four-variable Markov chain.

Now consider computing the marginal distribution of variable $x_3$.

Multiplication distributes over addition...

$P(x_3) = \sum_{x_1} \sum_{x_2} \sum_{x_4} P(x_1) P(x_2|x_1) P(x_3|x_2) P(x_4|x_3)$

$= \sum_{x_1} \sum_{x_2} P(x_1) P(x_2|x_1) \left( \sum_{x_4} P(x_4|x_3) \right)$

Can view as sending/combining messages...

Specific numerical example

Express marginals as product of messages

$p(x_i) = \frac{1}{Z} m_\alpha(x_i)m_\beta(x_i)$

Recursive evaluation of messages

$m_\alpha(x_i) = \sum_{x_{i-1}} \psi(x_{i-1}, x_i)m_\alpha(x_{i-1})$

$m_\beta(x_i) = \sum_{x_{i+1}} \psi(x_i, x_{i+1})m_\beta(x_{i+1})$

Find $Z$ by normalizing $p(x_i)$

Christopher Bishop, MSR
Specific numerical example

Joint Probability, represented in a truth table

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.1100</td>
</tr>
<tr>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.1100</td>
</tr>
<tr>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.0100</td>
</tr>
<tr>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.1049</td>
</tr>
<tr>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.0400</td>
</tr>
<tr>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.1000</td>
</tr>
<tr>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.0900</td>
</tr>
<tr>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.0900</td>
</tr>
<tr>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.0900</td>
</tr>
<tr>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>0.0900</td>
</tr>
</tbody>
</table>

Specific numerical example

Joint Probability

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Compute marginal of x3:

\[
P(x_3=1) = 0.458 \\
P(x_3=2) = 0.542
\]
Message Passing

Can view as sending/combining messages...

\[ p(x_i) = \frac{1}{Z} m_{\text{forward}}(x_i) m_{\text{backward}}(x_i) = \frac{[.458(1)] (.542)(1)]}{[.458 (.542)]} \]

These are the same values for the marginal \( P(x_3) \) that we computed from the raw joint probability table. Whew!!!

Specific numerical example

\[
\begin{align*}
P(\{x_1\}) &= \begin{bmatrix} .6 & .5 \end{bmatrix} \\
P(\{x_2\} | \{x_1\}) &= \begin{bmatrix} .4 & .6 \end{bmatrix} \\
P(\{x_3\} | \{x_1\}) &= \begin{bmatrix} .8 & .2 \end{bmatrix} \\
P(\{x_4\} | \{x_1\}) &= \begin{bmatrix} .5 & .5 \end{bmatrix}
\end{align*}
\]

If we want to compute all marginals, we can do it in one shot by cascading, for a big computational savings.

We need one cascaded forward pass, one separate cascaded backward pass, then a combination and normalization at each node.

Specific numerical example

\[
\begin{align*}
P(\{x_1\}) &= \begin{bmatrix} .6084 & .3916 \end{bmatrix} \\
\text{forward pass} &= \begin{bmatrix} [7.3] \end{bmatrix} \\
\text{backward pass} &= \begin{bmatrix} [.43 .57] \end{bmatrix} \\
\text{combined+ normalized} &= \begin{bmatrix} [.6084 .3916] \end{bmatrix}
\end{align*}
\]

Note: In this example, a directed Markov chain using true conditional probabilities (rows sum to one), only the forward pass really needed! This is true because the backward pass sums along rows, and always produces \([1 1]\)’.

We don’t really need forward AND backward in this example.

Specific numerical example

\[
\begin{align*}
P(\{x_1\}) &= \begin{bmatrix} .6084 & .3916 \end{bmatrix} \\
\text{forward pass} &= \begin{bmatrix} [7.3] \end{bmatrix} \\
\text{backward pass} &= \begin{bmatrix} [.43 .57] \end{bmatrix} \\
\text{combined+ normalized} &= \begin{bmatrix} [.6084 .3916] \end{bmatrix}
\end{align*}
\]

We don’t really need forward AND backward in this example.

Specific numerical example

\[
\begin{align*}
P(\{x_1\}) &= \begin{bmatrix} .6084 & .3916 \end{bmatrix} \\
\text{forward pass} &= \begin{bmatrix} [7.3] \end{bmatrix} \\
\text{backward pass} &= \begin{bmatrix} [.43 .57] \end{bmatrix} \\
\text{combined+ normalized} &= \begin{bmatrix} [.6084 .3916] \end{bmatrix}
\end{align*}
\]

These are already the marginals for this example.
Belief Propagation

- Extension to general tree-structured graphs
- Works for both directed AND undirected graphs
- At each node:
  - form product of incoming messages and local evidence
  - marginalize to give outgoing message
  - one message in each direction across every link

- Gives exact answer in any graph with no loops.

Loopy Belief Propagation

- BP applied to graph that contains loops
  - needs a propagation “schedule”
  - needs multiple iterations
  - might not converge

- Typically works well, even though it isn’t supposed to
- State-of-the-art performance in error-correcting codes

MAP Estimation

Sum-product algorithm lets us take a joint distribution \( P(X) \) and efficiently compute the marginal distribution over any components variable \( x_i \).

Another common task is to find the settings of variables that yield the largest probability. This is addressed using the max-product algorithm.

Max-product Algorithm

- Goal: find \( x_{MAP} = \arg \max_x P(x) \)
  - define \( \phi(x_i) = \max_{x_{i-1}} \max_{x_{i+1}} \cdots \max_{x_L} p(x_1, \ldots, x_L) \)
  - then \( x_{i,MAP} = \arg \max_{x_i} \phi(x_i) \)

- Message passing algorithm with “sum” replaced by “max”
- Generalizes to any two operations forming a semiring

Computing MAP Value

\[
\phi(x_i) = \max_{x_1} \cdots \max_{x_{i-1}} \max_{x_{i+1}} \cdots \max_{x_L} p(x_1, \ldots, x_L)
\]

Can solve using message passing algorithm with “sum” replaced by “max”.

In our chain, we start at the end and work our way back to the root (\( x_1 \)) using the max-product algorithm, keeping track of the max value as we go.

Computing Arg-Max of MAP Value

\[
x_{i,MAP} = \arg \max_{x_i} \phi(x_i)
\]

However, to recover the specific variable values that yield the MAP estimate (that is, argmax of the MAP), we have to essentially write a Dynamic Programming algorithm, based on max-product.
Specific numerical example: MAP Estimate

State Space Trellis

Specific numerical example: MAP Estimate

State Space Trellis

Specific numerical example

Joint Probability, represented in a truth table

Largest value of joint prob = mode = MAP achieved for

x1=1, x2=2, x3=2, x4=1