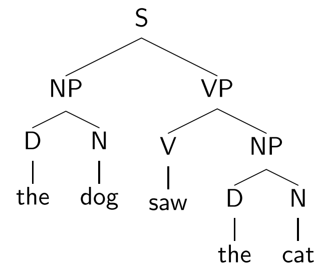
Lecture 4, COMS E6998-3: Discriminative Context-Free Parsing

Michael Collins

February 9, 2011

Context-Free Parse Trees



- Each rule is a tuple $\langle X \rightarrow Y\ Z, i, k, j \rangle$ where $X \rightarrow Y\ Z$ is a rule, non-terminal X spans words $i \ldots j$ inclusive, Y spans words $i \ldots k$ inclusive, Z spans words $(k+1) \ldots j$ inclusive.
- Rules in this example:

$$S \rightarrow NP\ VP, 1, 2, 5$$
$$NP \rightarrow D\ N, 1, 1, 2$$
$$VP \rightarrow V\ NP, 3, 3, 5$$
$$NP \rightarrow D\ N, 4, 4, 5$$

Context-Free Grammars

- A context-free grammar (CFG) in Chomsky normal form is a tuple (V, Σ, R, S) where:
  - V is a finite set of *non-terminal* symbols
  - Σ is a finite set of *terminal* symbols
  - R is a set of rules: each rule either takes the form

    $$X \rightarrow Y\ Z$$

    where $X, Y, Z \in V$, or

    $$X \rightarrow w$$

    where $X \in V$ and $w \in \Sigma$
  - $S \in V$ is the start symbol

Ambiguity

There are many sources of ambiguity: PP attachment, part-of-speech ambiguity, coordination, etc. etc.

Notation

- Assume $\underline{x}$ is a sequence of words $x_1 \ldots x_m$
- A context-free parse is a vector $\underline{y}$
- First, define the *index set* $\mathcal{I}$ to be the set of all possible rules. For example, for $m = 3$,

$$\mathcal{I} = \{X \rightarrow Y\ Z, i, k, j : X \rightarrow Y\ Z \in R, 1 \leq i \leq k < j \leq m\}$$

- Then $\underline{y}$ is a vector of values $y(r)$ for all $r \in \mathcal{I}$. $y(r) = 1$ if the structure contains the rule (r), $y(r) = 0$ otherwise.
- We use $\mathcal{Y}$ to refer to the set of all possible well-formed vectors $\underline{y}$

CRFs for Discriminative Context-Free Parsing

- We use $\underline{\Phi}(\underline{x}, \underline{y}) \in \mathbb{R}^d$ to refer to a feature vector for an *entire* dependency structure $\underline{y}$
- We then build a log-linear model, very similar to a CRF

$$p(\underline{y}|\underline{x};\underline{w}) = \frac{\exp\left(\underline{w} \cdot \underline{\Phi}(\underline{x}, \underline{y})\right)}{\sum_{\underline{y}' \in \mathcal{Y}} \exp\left(\underline{w} \cdot \underline{\Phi}(\underline{x}, \underline{y}')\right)}$$

- How do we define $\underline{\Phi}(\underline{x}, \underline{y})$? Answer:

$$\underline{\Phi}(\underline{x}, \underline{y}) = \sum_{r \in \mathcal{I}} y(r) \underline{\phi}(\underline{x}, r)$$

where $\underline{\phi}(\underline{x}, r)$ is the feature vector for rule r

Feature Vectors for Rules

- $\underline{\phi}(\underline{x}, X \rightarrow Y\ Z, i, k, j)$ is a feature vector representing rule

$$X \rightarrow Y\ Z, i, k, j$$

for sentence $\underline{x}$
- Example features:
  - Identity of the rule $X \rightarrow Y\ Z$
  - Identity of the rule $X \rightarrow Y\ Z$ in conjunction with words at the boundary points i, k, or j
  - etc. etc.

Decoding

- The decoding problem: find

$$\begin{aligned}
\arg\max_{\underline{y} \in \mathcal{Y}} p(\underline{y}|\underline{x};\underline{w}) &= \arg\max_{\underline{y} \in \mathcal{Y}} \frac{\exp\left(\underline{w} \cdot \underline{\Phi}(\underline{x}, \underline{y})\right)}{\sum_{\underline{y}' \in \mathcal{Y}} \exp\left(\underline{w} \cdot \underline{\Phi}(\underline{x}, \underline{y}')\right)} \\
&= \arg\max_{\underline{y} \in \mathcal{Y}} \exp\left(\underline{w} \cdot \underline{\Phi}(\underline{x}, \underline{y})\right) \\
&= \arg\max_{\underline{y} \in \mathcal{Y}} \underline{w} \cdot \underline{\Phi}(\underline{x}, \underline{y}) \\
&= \arg\max_{\underline{y} \in \mathcal{Y}} \underline{w} \cdot \sum_{r \in \mathcal{I}} y(r) \underline{\phi}(\underline{x}, r) \\
&= \arg\max_{\underline{y} \in \mathcal{Y}} \sum_{r \in \mathcal{I}} y(r) \left(\underline{w} \cdot \underline{\phi}(\underline{x}, r)\right)
\end{aligned}$$

- This problem can be solved using dynamic programming, in $O(m^3)$ time, where m is the length of the sentence

Decoding using the CKY Algorithm

- For convenience, define

$$\theta(r) = \underline{w} \cdot \underline{\phi}(\underline{x}, r)$$

The decoding problem is to find

$$\arg\max_{\underline{y} \in \mathcal{Y}} \sum_{r \in \mathcal{I}} y(r)\theta(r)$$

- Dynamic programming algorithm: define

$$\pi[X, i, j]$$

for $X \in N$, $1 \leq i \leq j \leq m$ to be the highest score for any subtree rooted in non-terminal X, spanning words $i \dots j$ inclusive

Parameter Estimation

- To estimate the parameters, we assume we have a set of n labeled examples, $\{(\underline{x}^i, \underline{y}^i)\}_{i=1}^n$. Each $\underline{x}^i$ is an input sequence $x_1^i \dots x_m^i$, each $\underline{y}^i$ is a context-free tree
- We then proceed in exactly the same way as for CRFs
- The *regularized log-likelihood function* is

$$L(\underline{w}) = \sum_{i=1}^{n} \log p(\underline{y}^i | \underline{x}^i; \underline{w}) - \frac{\lambda}{2} ||\underline{w}||^2$$

- The *parameter estimates* are

$$\underline{w}^* = \arg\max_{\underline{w} \in \mathbb{R}^d} \sum_{i=1}^{n} \log p(\underline{y}^i | \underline{x}^i; \underline{w}) - \frac{\lambda}{2} ||\underline{w}||^2$$

The gradient of $L(\underline{w})$ can again be calculated efficiently, using dynamic programming algorithms

Decoding using the CKY Algorithm (continued)

- Initialization: for $i = 1 \dots m$, $X \in G$, define $\pi[X, i, i] = 0$ if $X \rightarrow x_i$ is a valid rule, $-\infty$ otherwise. (Recall that x_i is the i'th word in the input sentence.)

- Recursive case: for $X \in G$, for $1 \leq i < j \leq n$,

$$\pi[X, i, j] = \max_{\substack{X \rightarrow Y\ Z \in R, \\ k \in \{i \dots j-1\}}} \left(\theta(X \rightarrow Y\ Z, i, k, j) + \pi[Y, i, k] + \pi[Z, k+1, j]\right)$$

- The highest scoring tree has score $\pi[S, 1, m]$. Backpointers can be used to recover the identity of the highest scoring tree.

Lecture 4, COMS E6998-3: The Structured Perceptron

Michael Collins

February 9, 2011

Conditional Random Fields (CRFs)

- Notation: for convenience we'll use $\underline{x}$ to refer to the sequence of input words, $x_1 \ldots x_m$, and $\underline{s}$ to refer to a sequence of possible states, $s_1 \ldots s_m$. The set of possible states is $\mathcal{S}$. We use $\mathcal{Y}$ to refer to the set of *all possible state sequences* (we have $|\mathcal{Y}| = |S|^m$).
- We're again going to build a model of

$$p(s_1 \ldots s_m | x_1 \ldots x_m) = p(\underline{s}|\underline{x})$$

CRFs (continued)

$$p(\underline{s}|\underline{x};\underline{w}) = \frac{\exp\left(\underline{w}\cdot\underline{\Phi}(\underline{x},\underline{s})\right)}{\sum_{\underline{s}'\in\mathcal{Y}} \exp\left(\underline{w}\cdot\underline{\Phi}(\underline{x},\underline{s}')\right)}$$

- How do we define $\underline{\Phi}(\underline{x},\underline{s})$? Answer:

$$\underline{\Phi}(\underline{x},\underline{s}) = \sum_{j=1}^{m} \underline{\phi}(\underline{x}, j, s_{j-1}, s_j)$$

where $\underline{\phi}(\underline{x}, j, s_{j-1}, s_j)$ are the same as the feature vectors used in MEMMs.

CRFs

- We use $\underline{\Phi}(\underline{x},\underline{s}) \in \mathbb{R}^d$ to refer to a feature vector for an *entire* state sequence
- We then build a *giant* log-linear model,

$$p(\underline{s}|\underline{x};\underline{w}) = \frac{\exp\left(\underline{w}\cdot\underline{\Phi}(\underline{x},\underline{s})\right)}{\sum_{\underline{s}'\in\mathcal{Y}} \exp\left(\underline{w}\cdot\underline{\Phi}(\underline{x},\underline{s}')\right)}$$

- The model is "giant" in the sense that: 1) the space of possible values for $\underline{s}$, i.e., $\mathcal{Y}$, is huge. 2) The normalization constant (denominator in the above expression) involves a sum over a huge number of possibilities (i.e., all members of $\mathcal{Y}$).

Decoding with CRFs

- The decoding problem: find

$$\begin{aligned}
\arg\max_{\underline{s}\in\mathcal{Y}} p(\underline{s}|\underline{x};\underline{w}) &= \arg\max_{\underline{s}\in\mathcal{Y}} \frac{\exp\left(\underline{w}\cdot\underline{\Phi}(\underline{x},\underline{s})\right)}{\sum_{\underline{s}'\in\mathcal{Y}} \exp\left(\underline{w}\cdot\underline{\Phi}(\underline{x},\underline{s}')\right)} \\
&= \arg\max_{\underline{s}\in\mathcal{Y}} \exp\left(\underline{w}\cdot\underline{\Phi}(\underline{x},\underline{s})\right) \\
&= \arg\max_{\underline{s}\in\mathcal{Y}} \underline{w}\cdot\underline{\Phi}(\underline{x},\underline{s}) \\
&= \arg\max_{\underline{s}\in\mathcal{Y}} \underline{w}\cdot\sum_{j=1}^{m}\underline{\phi}(\underline{x}, j, s_{j-1}, s_j) \\
&= \arg\max_{\underline{s}\in\mathcal{Y}} \sum_{j=1}^{m}\underline{w}\cdot\underline{\phi}(\underline{x}, j, s_{j-1}, s_j)
\end{aligned}$$

- Again, we can use the Viterbi algorithm...

The Viterbi Algorithm for CRFs

- Initialization: for $s \in \mathcal{S}$
$$\pi[1, s] = \underline{w} \cdot \underline{\phi}(\underline{x}, 1, s_0, s)$$
where s_0 is a special "initial" state.
- For $j = 2 \ldots m$, $s = 1 \ldots k$:
$$\pi[j, s] = \max_{s' \in \mathcal{S}} \left[\pi[j-1, s'] + \underline{w} \cdot \underline{\phi}(\underline{x}, j, s', s)\right]$$
- We then have
$$\max_{s_1 \ldots s_m} \sum_{j=1}^{m} \underline{w} \cdot \underline{\phi}(\underline{x}, j, s_{j-1}, s_j) = \max_s \pi[m, s]$$
- The algorithm runs in $O(mk^2)$ time. As before (see HMM lecture slides), we can use backpointers to recover the most likely sequence of states.

The Structured Perceptron

- Input: labeled examples, $\{(\underline{x}^i, \underline{s}^i)\}_{i=1}^n$.
- Initialization: $\underline{w} = \underline{0}$
- For $t = 1 \ldots T$, for $i = 1 \ldots n$:
  - Use the Viterbi algorithm to calculate
$$\underline{s}^* = \arg\max_{\underline{s} \in \mathcal{Y}} \ \underline{w} \cdot \underline{\Phi}(\underline{x}^i, \underline{s}) = \arg\max_{\underline{s} \in \mathcal{Y}} \ \sum_{j=1}^{m} \underline{w} \cdot \underline{\phi}(\underline{x}, j, s_{j-1}, s_j)$$
  - Updates:
$$\begin{aligned} \underline{w} &= \underline{w} + \underline{\Phi}(\underline{x}^i, \underline{s}^i) - \underline{\Phi}(\underline{x}^i, \underline{s}^*) \\ &= \underline{w} + \sum_{j=1}^{m} \underline{\phi}(\underline{x}, j, s^i_{j-1}, s^i_j) - \sum_{j=1}^{m} \underline{\phi}(\underline{x}, j, s^*_{j-1}, s^*_j) \end{aligned}$$
- Return $\underline{w}$

Parameter Estimation in CRFs

- To estimate the parameters, we assume we have a set of n labeled examples, $\{(\underline{x}^i, \underline{s}^i)\}_{i=1}^n$. Each $\underline{x}^i$ is an input sequence $x^i_1 \ldots x^i_m$, each $\underline{s}^i$ is a state sequence $s^i_1 \ldots s^i_m$.
- We then proceed in exactly the same way as for regular log-linear models
- The *regularized log-likelihood function* is
$$L(\underline{w}) = \sum_{i=1}^{n} \log p(\underline{s}^i | \underline{x}^i; \underline{w}) - \frac{\lambda}{2} ||\underline{w}||^2$$
- Our parameter estimates are
$$\underline{w}^* = \arg\max_{\underline{w} \in \mathbb{R}^d} \ \sum_{i=1}^{n} \log p(\underline{s}^i | \underline{x}^i; \underline{w}) - \frac{\lambda}{2} ||\underline{w}||^2$$
- We find the optimal parameters using gradient-based methods

The Structured Perceptron with Averaging

- Input: labeled examples, $\{(\underline{x}^i, \underline{s}^i)\}_{i=1}^n$.
Initialization: $\underline{w} = \underline{0}$, $\underline{w}_a = \underline{0}$
- For $t = 1 \ldots T$, for $i = 1 \ldots n$:
  - Use the Viterbi algorithm to calculate
$$\underline{s}^* = \arg\max_{\underline{s} \in \mathcal{Y}} \ \underline{w} \cdot \underline{\Phi}(\underline{x}^i, \underline{s}) = \arg\max_{\underline{s} \in \mathcal{Y}} \ \sum_{j=1}^{m} \underline{w} \cdot \underline{\phi}(\underline{x}, j, s_{j-1}, s_j)$$
  - Updates:
$$\begin{aligned} \underline{w} &= \underline{w} + \underline{\Phi}(\underline{x}^i, \underline{s}^i) - \underline{\Phi}(\underline{x}^i, \underline{s}^*) \\ &= \underline{w} + \sum_{j=1}^{m} \underline{\phi}(\underline{x}, j, s^i_{j-1}, s^i_j) - \sum_{j=1}^{m} \underline{\phi}(\underline{x}, j, s^*_{j-1}, s^*_j) \\ \underline{w}_a &= \underline{w}_a + \underline{w} \end{aligned}$$
- Return $\underline{w}_a / nT$

Convergence of the Structured Perceptron

- **Definition:** The training set $\{(\underline{x}^i, \underline{s}^i)\}_{i=1}^n$ is separable with margin $\delta > 0$, if there exists some parameter vector $\underline{w}$ such that:
  1. $||\underline{w}||^2 = 1$
  2. For all $i = 1 \ldots n$, for all $s_1 \ldots s_m$ such that $s_j \neq s_j^i$ for some j,
  $$\underline{w} \cdot \underline{\Phi}(\underline{x}^i, \underline{s}^i) - \underline{w} \cdot \underline{\Phi}(\underline{x}^i, \underline{s}) \geq \delta$$
- **Theorem:** if a training set is separable with margin δ, the structured perceptron makes at most
$$\frac{R^2}{\delta^2}$$
mistakes before convergence, where R is related to the norm of the feature vectors $\underline{\Phi}(\underline{x}^i, \underline{s})$

Conditional Random Fields (CRFs)

- Notation: for convenience we'll use $\underline{x}$ to refer to the sequence of input words, $x_1 \ldots x_m$, and $\underline{s}$ to refer to a sequence of possible states, $s_1 \ldots s_m$. The set of possible states is $\mathcal{S}$. We use $\mathcal{S}^m$ to refer to the set of *all possible state sequences* (we have $|\mathcal{S}^m| = |S|^m$).
- We're again going to build a model of
$$p(s_1 \ldots s_m | x_1 \ldots x_m) = p(\underline{s}|\underline{x})$$

Lecture 4, COMS E6998-3: Pairwise CRFs

Michael Collins

February 9, 2011

CRFs

- We use $\underline{\Phi}(\underline{x}, \underline{s}) \in \mathbb{R}^d$ to refer to a feature vector for an *entire* state sequence
- We then build a *giant* log-linear model,
$$p(\underline{s}|\underline{x}; \underline{w}) = \frac{\exp\left(\underline{w} \cdot \underline{\Phi}(\underline{x}, \underline{s})\right)}{\sum_{\underline{s}' \in \mathcal{S}^m} \exp\left(\underline{w} \cdot \underline{\Phi}(\underline{x}, \underline{s}')\right)}$$
- The model is "giant" in the sense that: 1) the space of possible values for $\underline{s}$, i.e., $\mathcal{S}^m$, is huge. 2) The normalization constant (denominator in the above expression) involves a sum over a huge number of possibilities (i.e., all members of $\mathcal{S}^m$).

CRFs (continued)

$$p(\underline{s}|\underline{x};\underline{w}) = \frac{\exp\left(\underline{w}\cdot\underline{\Phi}(\underline{x},\underline{s})\right)}{\sum_{\underline{s}'\in\mathcal{S}^m}\exp\left(\underline{w}\cdot\underline{\Phi}(\underline{x},\underline{s}')\right)}$$

- We assume that we have a graph with nodes $V = \{1, 2, \ldots, m\}$, and a set of undirected edges E
- How do we define $\underline{\Phi}(\underline{x},\underline{s})$? Answer:

$$\underline{\Phi}(\underline{x},\underline{s}) = \sum_{(j,k)\in E} \underline{\phi}(\underline{x}, j, k, s_j, s_k)$$

Parameter Estimation in CRFs

- To estimate the parameters, we assume we have a set of n labeled examples, $\{(\underline{x}^i, \underline{s}^i)\}_{i=1}^n$. Each $\underline{x}^i$ is an input sequence $x_1^i \ldots x_m^i$, each $\underline{s}^i$ is a state sequence $s_1^i \ldots s_m^i$.
- We then proceed in exactly the same way as for regular log-linear models
- The *regularized log-likelihood function* is

$$L(\underline{w}) = \sum_{i=1}^n \log p(\underline{s}^i|\underline{x}^i;\underline{w}) - \frac{\lambda}{2}||\underline{w}||^2$$

- Our parameter estimates are

$$\underline{w}^* = \arg\max_{\underline{w}\in\mathbb{R}^d} \sum_{i=1}^n \log p(\underline{s}^i|\underline{x}^i;\underline{w}) - \frac{\lambda}{2}||\underline{w}||^2$$

Decoding with CRFs

- The decoding problem: find

$$\begin{aligned}
\arg\max_{\underline{s}\in\mathcal{S}^m} p(\underline{s}|\underline{x};\underline{w}) &= \arg\max_{\underline{s}\in\mathcal{S}^m} \frac{\exp\left(\underline{w}\cdot\underline{\Phi}(\underline{x},\underline{s})\right)}{\sum_{\underline{s}'\in\mathcal{S}^m}\exp\left(\underline{w}\cdot\underline{\Phi}(\underline{x},\underline{s}')\right)} \\
&= \arg\max_{\underline{s}\in\mathcal{S}^m} \exp\left(\underline{w}\cdot\underline{\Phi}(\underline{x},\underline{s})\right) \\
&= \arg\max_{\underline{s}\in\mathcal{S}^m} \underline{w}\cdot\underline{\Phi}(\underline{x},\underline{s}) \\
&= \arg\max_{\underline{s}\in\mathcal{S}^m} \underline{w}\cdot\sum_{(j,k)\in E}\underline{\phi}(\underline{x}, j, k, s_j, s_k) \\
&= \arg\max_{\underline{s}\in\mathcal{S}^m} \sum_{(j,k)\in E}\theta(j, k, s_j, s_k)
\end{aligned}$$

where

$$\theta(j, k, s_j, s_k) = \underline{w}\cdot\underline{\phi}(\underline{x}, j, k, s_j, s_k)$$

Finding the Maximum-Likelihood Estimates

- We'll again use gradient-based optimization methods to find $\underline{w}^*$
- How can we compute the derivatives? As before,

$$\frac{\partial}{\partial w_l}L(\underline{w}) = \sum_i \Phi_l(\underline{x}^i,\underline{s}^i) - \sum_i\sum_{\underline{s}\in\mathcal{S}^m} p(\underline{s}|\underline{x}^i;\underline{w})\Phi_l(\underline{x}^i,\underline{s}) - \lambda w_l$$

- The first term is easily computed, because

$$\sum_i \Phi_l(\underline{x}^i,\underline{s}^i) = \sum_i \sum_{(j,k)\in E} \phi_l(\underline{x}^i, j, k, s_j^i, s_k^i)$$

- The second term involves a sum over $\mathcal{S}^m$, and because of this looks nasty...

Calculating Derivatives using the Forward-Backward Algorithm

- We now consider how to compute the second term:

$$
\begin{aligned}
\sum_{\underline{s} \in \mathcal{S}^m} p(\underline{s}|\underline{x}^i;\underline{w})\Phi_l(\underline{x}^i,\underline{s}) &= \sum_{\underline{s} \in \mathcal{S}^m} p(\underline{s}|\underline{x}^i;\underline{w}) \sum_{(j,k)\in E} \phi_l(\underline{x}^i, j, k, s_j, s_k) \\
&= \sum_{(j,k)\in E} \sum_{a\in\mathcal{S}, b\in\mathcal{S}} q^i_{j,k}(a,b)\phi_l(\underline{x}^i, j, k, a, b)
\end{aligned}
$$

where

$$
q^i_{j,k}(a,b) = \sum_{\underline{s}\in\mathcal{S}^m : s_j = a, s_k = b} p(\underline{s}|\underline{x}^i;\underline{w})
$$

The Model Form for Markov Random Fields

- Model form for a pairwise MRF

$$
\begin{aligned}
p(x_1, x_2, \ldots x_n; \underline{\Theta}) &= \frac{1}{Z(\underline{\theta})} \exp\{\sum_{(i,j)\in E} \theta_{i,j}(x_i, x_j)\} \\
&= \frac{1}{Z(\underline{\theta})} \prod_{(i,j)\in E} \psi_{i,j}(x_i, x_j)
\end{aligned}
$$

where

- E is the set of edges in the undirected graph
- $\psi_{i,j}(x_i, x_j) = \exp\{\theta_{i,j}(x_i, x_j)\}$
- $Z(\underline{\theta}) = \sum_{x_1 \ldots x_n} \prod_{(i,j)\in E} \psi_{i,j}(x_i, x_j)$

Lecture 4: Belief Propagation

Michael Collins

February 9, 2011

Two Key Problems

- (1) Computing the partition function,

$$
Z(\underline{\theta}) = \sum_{x_1 \ldots x_n} \prod_{(i,j)\in E} \psi_{i,j}(x_i, x_j)
$$

- (2) Computing marginal probabilities under the model,

$$
P(X_i = x; \underline{\theta})
$$

for any random variable X_i, for any value x. e.g., $P(X_7 = +1; \underline{\theta})$

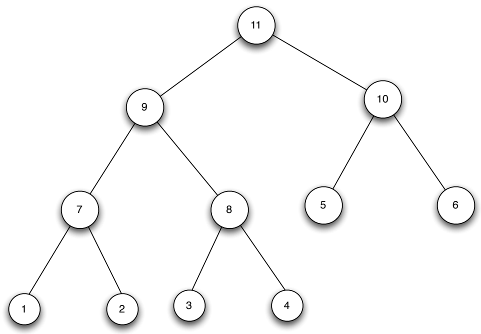
- Note:

$$
P(X_i = x; \underline{\theta}) = \frac{1}{Z(\underline{\theta})} \sum_{x_1 \ldots x_n} \delta(x_i, x) \prod_{(i,j)\in E} \psi_{i,j}(x_i, x_j)
$$

where $\delta(x_i, x) = 1$ if $x_i = x$, 0 otherwise

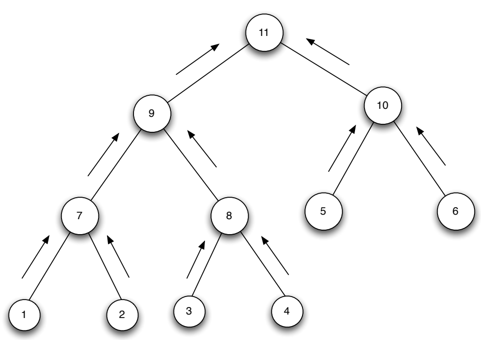
Tree-Structured MRFs

- In this lecture, we'll consider the case where the underlying graph is a tree (easy case: dynamic programming)
- First step: pick one of the vertices in the tree as a root (any node will do). In this example, we pick node 11:



Tree-Structured MRFs

- In a first step, we'll send "messages" up through the tree
- The messages basically correspond to bottom-up dynamic programming



Computing the Partition Function

For each possible value of x_7, calculate

$$m_{1\to 7}(x_7) = \sum_{x_1} \psi_{1,7}(x_1, x_7)$$

$m_{1\to 7}(x_7)$ is a "message" from node 1 to node 7 about the possible value x_7 for node 7.
Similarly, calculate

$$m_{2\to 7}(x_7) = \sum_{x_2} \psi_{2,7}(x_2, x_7) \quad m_{3\to 8}(x_8) = \sum_{x_3} \psi_{3,8}(x_3, x_8)$$

$$m_{4\to 8}(x_8) = \sum_{x_4} \psi_{4,8}(x_4, x_8) \quad m_{5\to 10}(x_{10}) = \sum_{x_5} \psi_{5,10}(x_5, x_{10})$$

$$m_{6\to 10}(x_{10}) = \sum_{x_6} \psi_{6,10}(x_6, x_{10})$$

The General Form for the Messages

- For any node i, define $N(i)$ to be the set of neighbors of i in the graph:
$$N(i) = \{j : (i,j) \in E\}$$
- The messages are then defined as
$$m_{i\to j}(x_j) = \sum_{x_i} \psi_{i,j}(x_i, x_j) \prod_{k\in N(i), k\neq j} m_{k\to i}(x_i)$$
- Note: special case, if $N(i) = \{j\}$, then message is
$$m_{i\to j}(x_j) = \sum_{x_i} \psi_{i,j}(x_i, x_j)$$

Computing the Partition Function

Next, calculate

$$m_{7\to 9}(x_9) = \sum_{x_7} \psi_{7,9}(x_7, x_9) m_{1\to 7}(x_7) m_{2\to 7}(x_7)$$

$$m_{8\to 9}(x_9) = \sum_{x_8} \psi_{8,9}(x_8, x_9) m_{3\to 8}(x_8) m_{4\to 8}(x_8)$$

$$m_{9\to 11}(x_{11}) = \sum_{x_9} \psi_{9,11}(x_9, x_{11}) m_{7\to 9}(x_9) m_{8\to 9}(x_9)$$

$$m_{10\to 11}(x_{11}) = \sum_{x_{10}} \psi_{10,11}(x_{10}, x_{11}) m_{5\to 10}(x_{10}) m_{6\to 10}(x_{10})$$

And finally,

$$Z(\underline{\theta}) = \sum_{x_{11}} m_{9\to 11}(x_{11}) m_{10\to 11}(x_{11})$$

Two Key Problems

- (1) Computing the partition function,

$$Z(\underline{\theta}) = \sum_{x_1 \ldots x_n} \prod_{(i,j)\in E} \psi_{i,j}(x_i, x_j)$$

- (2) Computing marginal probabilities under the model,

$$P(X_i = x; \underline{\theta})$$

  for any random variable X_i, for any value x. e.g., $P(X_7 = +1; \underline{\theta})$

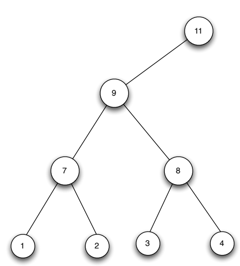
- Note:

$$P(X_i = x; \underline{\theta}) = \frac{1}{Z(\underline{\theta})} \sum_{x_1 \ldots x_n} \delta(x_i, x) \prod_{(i,j)\in E} \psi_{i,j}(x_i, x_j)$$

  where $\delta(x_i, x) = 1$ if $x_i = x$, 0 otherwise

What do the Messages Represent?

- An example: $m_{9\to 11}(x_{11})$
- Take $T(9, 11)$ to be the following subtree:

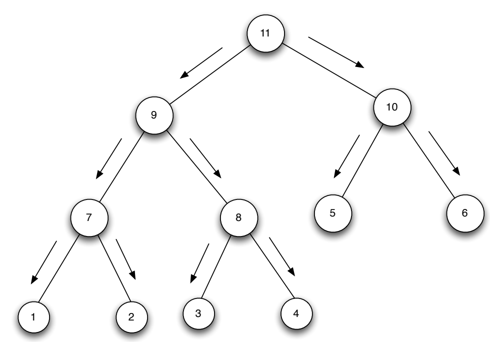


  The subtree contains the edge $(9, 11)$, together with the subtree rooted at node 9 that is the component when the edge $(9, 11)$ is removed from the graph

- Then $m_{9\to 11}(x_{11}) = \sum_{x_1,x_2,x_3,x_4,x_7,x_8,x_9} \prod_{(i,j)\in T(9,11)} \psi_{i,j}(x_i, x_j)$

Belief Propagation (Continued)

- In a second step, we'll send "messages" **down** the tree



- We use the same definition as before,

$$m_{i\to j}(x_j) = \sum_{x_i} \psi_{i,j}(x_i, x_j) \prod_{k\in N(i), k\neq j} m_{k\to i}(x_i)$$

Downward Messages

$$m_{11\to 9}(x_9) = \sum_{x_{11}} \psi_{11,9}(x_{11}, x_9) m_{10\to 11}(x_{11})$$

$$m_{11\to 10}(x_{10}) = \sum_{x_{11}} \psi_{11,10}(x_{11}, x_{10}) m_{9\to 11}(x_{11})$$

$$m_{9\to 7}(x_7) = \sum_{x_9} \psi_{9,7}(x_9, x_7) m_{11\to 9}(x_9) m_{8\to 9}(x_9)$$

$$m_{9\to 8}(x_8) = \sum_{x_9} \psi_{9,8}(x_9, x_8) m_{11\to 9}(x_9) m_{7\to 9}(x_9)$$

and so on, until all the downward messages are computed

Summary

- We choose one node of the tree as the root (in our example, we chose node 11)
- First compute messages $m_{i\to j}(x_j)$ bottom-up in the tree
- Then compute messages top-down through the tree: at this point we have messages between all pairs of nodes, in both directions
- We can then take the root messages, and calculate the partition function, e.g., $Z(\underline{\theta}) = \sum_{x_{11}} m_{9\to 11}(x_{11}) m_{10\to 11}(x_{11})$
- And we can also compute the full set of marginal probabilities

$$P(X_i = x; \underline{\theta}) = \frac{1}{Z(\underline{\theta})} \prod_{j\in N(i)} m_{j\to i}(x)$$

Computing Marginals

- Once we have all the messages, we can easily compute marginals
- For example,

$$P(X_9 = +1; \underline{\theta}) = \frac{1}{Z(\underline{\theta})} m_{7\to 9}(+1) m_{8\to 9}(+1) m_{11\to 9}(+1)$$

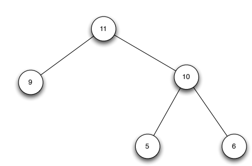
- The general form:

$$P(X_i = x; \underline{\theta}) = \frac{1}{Z(\underline{\theta})} \prod_{j\in N(i)} m_{j\to i}(x)$$

where $N(i) = \{j : (i,j) \in E\}$

What do the Messages Represent?

- A second example: $m_{11\to 9}(x_9)$
- Take $T(11, 9)$ to be the following subtree:



The subtree contains the edge $(11, 9)$, together with the subtree rooted at node 11 that is the component when the edge $(11, 9)$ is removed from the graph

- Then $m_{11\to 9}(x_9) = \sum_{x_5, x_6, x_{10}, x_{11}} \prod_{(i,j)\in T(11,9)} \psi_{i,j}(x_i, x_j)$

Finding the Maximum

- How do we find $\max_{x_1 \ldots x_n} \prod_{(i,j) \in E} \psi_{i,j}(x_i, x_j)$?
- For any node i, define $N(i)$ to be the set of neighbors of i in the graph:

$$N(i) = \{j : (i,j) \in E\}$$

- The messages are then defined as

$$m_{i \to j}(x_j) = \max_{x_i} \psi_{i,j}(x_i, x_j) \prod_{k \in N(i), k \neq j} m_{k \to i}(x_i)$$

- Note: special case, if $N(i) = \{j\}$, then message is $m_{i \to j}(x_j) = \max_{x_i} \psi_{i,j}(x_i, x_j)$
- And finally (for our example), the maximum scoring assignment is

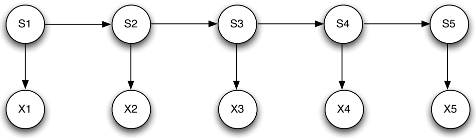
$$\max_{x_{11}} m_{9 \to 11}(x_{11}) m_{10 \to 11}(x_{11})$$

Hidden Markov Models (HMMs)

- In HMMs, we assume that:

$$P(X_1 = x_1, \ldots, X_m = x_m, S_1 = s_1, \ldots, S_m = s_m)$$
$$= P(S_1 = s_1) \prod_{j=2}^{m} P(S_j = s_j | S_{j-1} = s_{j-1}) \prod_{j=1}^{m} P(X_j = x_j | S_j = s_j)$$

- A Bayesian network representing the HMM (assume $m = 5$):



- If we run belief propogation on this Bayesian network, we recover the forward-backward algorithm