What is a Conditional Random Field?

Before we describe a CRF informally as an MRF where the $x$ variables are always observed,

$$p(x, y) = p(x)p(y | x)$$

Here's a better definition. A CRF defines an MRF over $y$ for every fixed value of $x$:

$$p(y | x) = \frac{1}{Z(x)} \prod_{c \in C} \phi_c(x, y_c), \quad Z(x) = \sum_y \prod_{c \in C} \phi_c(x, y_c)$$

Notes:

- No distribution over $x$  
  $p(x)$
- Normalized separately for each $x$
- Each potential $\phi_c$ can depend arbitrarily on $x$ (often designed with “local” connections to selected entries of $x$, but not necessary)
- Cliques $c$ are subsets of the $y$ indices
Learning in CRFs

\( \text{MRF: } p(x, y) \)

In CRFs, we maximize the conditional log-likelihood:

\[
\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p(y^{(n)} | x^{(n)})
\]

Some aspects are similar to learning in MRFs. A key difference is that the “model marginals” are different for each data case, because the normalization constant \( Z(x^{(n)}) \) is different.

"data marginal" vs. "model marginal"

(see HW2, HW3)

Example: Logistic Regression

Logistic regression is a simple CRF with \( y \in \{0, 1\} \).

\[
\log p(y | x) = \frac{1}{\text{\( Z(x) \)}} \left( \exp(\theta^T x \cdot I[y = 1]) \right)
\]

\[
Z(x) = \exp(\theta^T x) + 1
\]

\[
p_{\theta}(y = 1 | x) = \frac{\exp(\theta^T x)}{1 + \exp(\theta^T x)} = \text{sigmoid}(\theta^T x)
\]

Example: Chain CRF

One way to view a chain-structured CRF is as a sequence of logistic regression models, with pairwise connections between adjacent \( y \) variables to encourage a particular sequential structure in predicted labels:
What is a Conditional Random Field?
Message-Passing Implementation
Message Passing in Trees
Discussion and Extensions

Message-Passing Implementation

Overflow/Underflow and Log-Sum-Exp

- When factor values are small or large, or with many factors, messages can underflow or overflow since they are products of many terms. A common solution is to manipulate all factors and messages in log space.

- Example: consider the common factor manipulation

\[
\alpha(x) = \log \sum_y B(x, y) C(y)
\]

Let's compute \(\alpha(x) = \log \alpha(x)\) from \(\beta(x, y) = \log B(x, y)\) and \(\gamma(y) = \log C(y)\)

- Step 1: multiplication of factors is addition of log-factors

\[
\lambda(x, y) := \log(B(x, y)C(y)) = \beta(x, y) + \gamma(y)
\]

Step 2: marginalization requires exponentiation ("log-sum-exp")

\[
\alpha(x) = \log \left( \sum_y \exp \lambda(x, y) \right)
\]

\(\log - \text{sum} - \exp \)

Numerically Stable log-sum-exp

Before exponentiating, we need to be careful to shift values to avoid overflow/underflow

\[
\log \sum_k \exp(a_k)
\]

- \(c \leftarrow \max_i a_i\)
- return \(c + \log \sum_i \exp(a_i - c)\)

See `scipy.special.logsumexp`

(Comment: log-space implementation probably not needed in HW2, probably needed in HW3.)
What is a Conditional Random Field?

Message-Passing Implementation

Message Passing in Trees

Discussion and Extensions

Message Passing in Trees

A more general version of message passing works for any tree-structured MRF, that is, an MRF of the following form where $G = (V, E)$ is a tree:

$p(x) = \prod_{i \in V} \phi_i(x_i) \prod_{(i, j) \in E} \phi_{ij}(x_i, x_j)$.

By similar reasoning, the pairwise marginal for $(i, j) \in E$ is

$p(x_i, x_j) = \frac{1}{Z} \phi_i(x_i) \phi_{ij}(x_i, x_j) \phi_j(x_j) \prod_{k \in \text{nb}(i) \setminus j} m_{k \rightarrow i}(x_k) \prod_{\ell \in \text{nb}(j) \setminus i} m_{\ell \rightarrow j}(x_\ell)$.

The “message” $m_{j \rightarrow i}(x_i)$ is the result of summing out all factors and variables in the subtree $T_j$ rooted at $x_j$. 

Message passing can be derived from variable elimination. Take $x_i$ as the root and eliminate variables from leaf to root. We get

$Z = \sum_{x_i} \phi_i(x_i) \prod_{j \in \text{nb}(i)} m_{j \rightarrow i}(x_i)$

$p(x_i) = \frac{1}{Z} \phi_i(x_i) \prod_{j \in \text{nb}(i)} m_{j \rightarrow i}(x_i)$

The “message” $m_{j \rightarrow i}(x_i)$ is the result of summing out all factors and variables in the subtree $T_j$ rooted at $x_j$. 

Message Passing in Trees

Discussion and Extensions

13 / 24

14 / 24

15 / 24

16 / 24
Recurrence for Messages

The messages satisfy the following recurrence:

\[ m_{j \rightarrow i}(x_i) = \sum_{x_j} \phi_j(x_j) \phi_{ij}(x_i, x_j) \prod_{k \in nb(j) \setminus i} m_{k \rightarrow j}(x_j) \]

This can be understood by expanding the summation over \( T_j \) to group factors for subtrees rooted at each child of \( x_j \), that is, for each node \( k \in nb(j) \setminus i \).

Message-Passing Summary

\[
m_{j \rightarrow i}(x_i) = \sum_{x_j} \phi_j(x_j) \phi_{ij}(x_i, x_j) \prod_{k \in nb(j) \setminus i} m_{k \rightarrow j}(x_j)
\]

\[
Z = \sum_{x_i} \phi_i(x_i) \prod_{j \in nb(i)} m_{j \rightarrow i}(x_i)
\]

\[
p(x_i) = \frac{1}{Z} \phi_i(x_i) \prod_{j \in nb(i)} m_{j \rightarrow i}(x_i) \quad \forall i
\]

\[
p(x, x_j) = \frac{1}{Z} \phi_i(x_i) \phi_j(x_i, x_j) \hat{\phi}_{ij}(x_i, x_j) \prod_{k \in nb(i) \setminus j} m_{k \rightarrow i}(x_i) \prod_{t \in nb(j) \setminus i} m_{t \rightarrow j}(x_j) \quad (i, j) \in E
\]

Message-Passing

Importantly, the message from \( j \) to \( i \) doesn’t depend on which particular node is the root. There are only \( 2(n-1) \) total messages and we can compute them all in two passes through the tree.

Say that \( j \) is **ready to send** to \( i \) if \( j \) has received messages from all \( k \in nb(j) \setminus i \).

**Message passing**: while any node \( j \) is ready to send to \( i \), compute \( m_{j \rightarrow i} \) as

\[
m_{j \rightarrow i}(x_i) = \sum_{x_j} \phi_j(x_j) \phi_{ij}(x_i, x_j) \prod_{k \in nb(j) \setminus i} m_{k \rightarrow j}(x_j)
\]

This algorithm is described asynchronously (“ready-to-send”), but in practice: pass messages from leaves to root of tree and back.
What is a Conditional Random Field?

Message-Passing Implementation

Message Passing in Trees

Discussion and Extensions

Discussion

- Message-passing computes all single and pairwise marginals at roughly 2x cost of variable elimination
- It is restricted to pairwise MRFs and trees, but can be extended in some ways
- For exactly answering one query in any MRF, variable elimination is faster than message passing
- For exactly answering a set of marginal queries, variable elimination usually takes at most a factor of $O(n)$ more time

Sketches of Extensions

- What if the MRF has factors on more than two variables? (keyword: factor graphs)
- What if the MRF is not tree-structured, i.e., $G$ has cycles?
  - **Answer 1**: group nodes (keyword: clique trees or junction trees)
  - **Answer 2**: use message-passing as a fixed-point iteration (keyword: loopy belief propagation)