ECE521 Lecture 21

HMM cont.

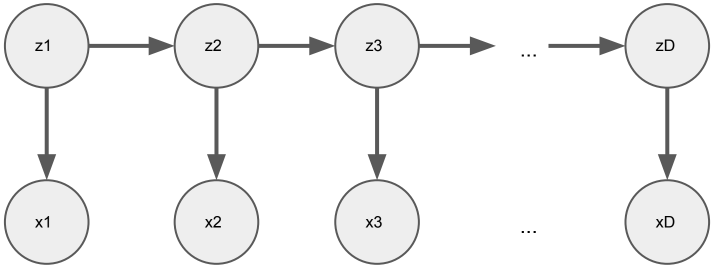
Message Passing Algorithms

UNIVERSITY OF TORONTO

Outline

- **Hidden Markov models**
  - Numerical example of figuring out marginal of the observed sequence
  - Numerical example of figuring out the most probable sequence
- Message-passing algorithm
  - Factor graph review
  - Sum-product algorithms

Inference in HMMs



- Given the joint distribution over the observations and the latent states:

$$p(x_1, \ldots, x_D, z_1, \ldots, z_D) = p(z_1) \prod_{t=2}^{D} p(z_t|z_{t-1}) \prod_{t=1}^{D} p(x_t|z_t)$$

  - Computing the marginal distribution of the observations requires sum over the latent states:

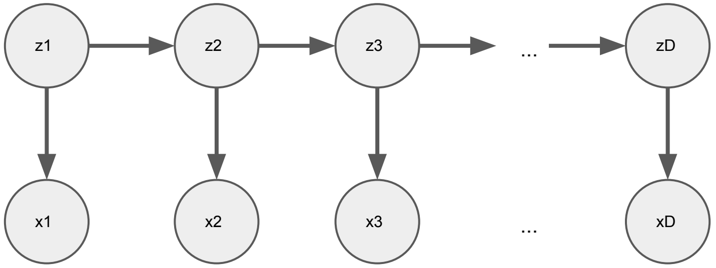
$$p(x_1, \ldots, x_D) = \sum_{z_1, \ldots, z_D} p(z_1) \prod_{t=2}^{D} p(z_t|z_{t-1}) \prod_{t=1}^{D} p(x_t|z_t)$$

$$= \underbrace{\sum_{z_D} \Big\{ \ldots \Big\{ \sum_{z_2} \Big\{ \sum_{z_1}}_{\textbf{local summation}} p(z_1)p(x_1|z_1)p(z_2|z_1) \Big\} p(x_2|z_2)p(z_3|z_2) \Big\} \ldots p(z_D|z_{D-1}) \Big\} p(x_D|z_D)$$

  - The marginal dist. is used for both posterior inference and learning:

$$p(z_1, \ldots, z_D | x_1, \ldots, x_D) = \frac{p(x_1, \ldots, x_D, z_1, \ldots, z_D)}{p(x_1, \ldots, x_D)}$$

Inference in HMMs



- Example1:
  - Consider an HMM with three discrete states: $z_t \in \{1, 2, 3\}$
  - The observations are also discrete random variables: $x_t \in \{A, C, G, T\}$
  - We have the following transition and emission probabilities:

$p(z_1) = [0.5, 0.5, 0.]^T$

$p(z_t \mid z_{t-1})$	$z_t = 1$	2	3
$z_{t-1} = 1$	0.5	0.5	0
2	0.1	0.8	0.1
3	0	0.5	0.5

$p(x_t \mid z_t)$	$x_t =$ A	C	G	T
$z_t = 1$	0.3	0.1	0	0.6
2	0	0.8	0.2	0
3	0	0.1	0.9	0

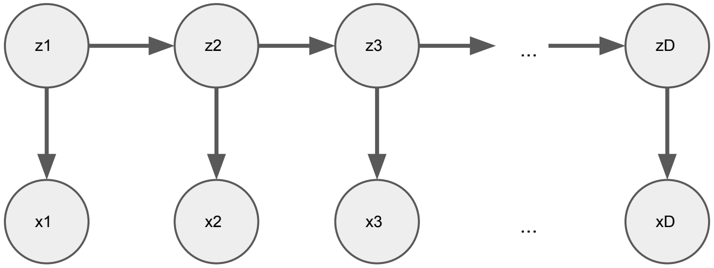
For a particular training example:

observations: $\underset{x_1}{\text{C}}\ \underset{x_2}{\text{C}}$

The marginal distribution of the observed sequence:

$$p(x_1 = C, x_2 = C) = \sum_{z_2} \left\{ \sum_{z_1} p(z_1) p(z_2|z_1) p(x_1 = C|z_1) \right\} p(x_2 = C|z_2)$$

Inference in HMMs



- Example1:
  - The marginal distribution of the observed sequence:

observations: $\underset{x_1}{\text{C}}\ \underset{x_2}{\text{C}}$

$$p(x_1 = C, x_2 = C) = \sum_{z_2} \left\{ \sum_{z_1} p(z_1) p(z_2|z_1) p(x_1 = C|z_1) \right\} p(x_2 = C|z_2)$$

$$= \sum_{z_2} \Big\{ p(z_1 = 1) p(z_2|z_1 = 1) p(x_1 = C|z_1 = 1) + p(z_1 = 2) p(z_2|z_1 = 2) p(x_1 = C|z_1 = 2)$$

$$+ p(z_1 = 3) p(z_2|z_1 = 3) p(x_1 = C|z_1 = 3) \Big\} p(x_2 = C|z_2)$$

$$= \sum_{z_2} \left\{ 0.5 \times \begin{bmatrix} 0.5 \\ 0.5 \\ 0 \end{bmatrix} \times 0.1 + 0.5 \times \begin{bmatrix} 0.1 \\ 0.8 \\ 0.1 \end{bmatrix} \times 0.8 + 0. \times \begin{bmatrix} 0.1 \\ 0.8 \\ 0.1 \end{bmatrix} \times 0.1 \right\} \times \begin{bmatrix} 0.1 \\ 0.8 \\ 0.1 \end{bmatrix}$$

$$= 0.29$$

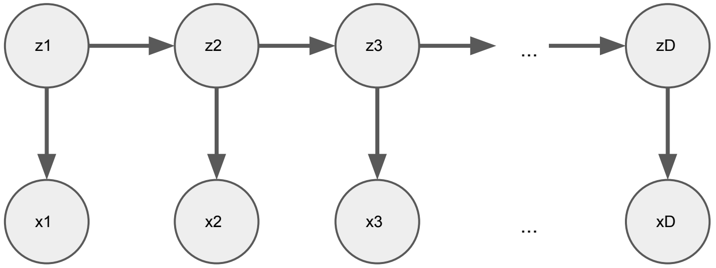
element-wise product

$p(z_1) = [0.5, 0.5, 0.]^T$

$p(z_t \mid z_{t-1})$	$z_t = 1$	2	3
$z_{t-1} = 1$	0.5	0.5	0
2	0.1	0.8	0.1
3	0	0.5	0.5

$p(x_t \mid z_t)$	$x_t =$ A	C	G	T
$z_t = 1$	0.3	0.1	0	0.6
2	0	0.8	0.2	0
3	0	0.1	0.9	0

Inference in HMMs



- Example2:
  - The posterior distribution over the latent states for a training example

observations: C C

x_1 x_2

$$p(z_1, z_2|x_1 = C, x_2 = C) = \frac{p(z_1)p(z_2|z_1)p(x_1 = C|z_1)p(x_2 = C|z_2)}{p(x_1 = C, x_2 = C)}$$

$$= \begin{bmatrix} 0.0025 & 0.02 & 0. \\ 0.004 & 0.256 & 0.004 \\ 0. & 0. & 0. \end{bmatrix} /p(x_1 = C, x_2 = C)$$

$$= \begin{bmatrix} 0.0025 & 0.02 & 0. \\ 0.004 & 0.256 & 0.004 \\ 0. & 0. & 0. \end{bmatrix} /0.2865$$

$$= \begin{bmatrix} 0.008726 & 0.07 & 0. \\ 0.014 & 0.89 & 0.014 \\ 0. & 0. & 0. \end{bmatrix}$$

$p(z_1) = [0.5, 0.5, 0.]^T$

$p(z_t\|z_{t-1})$	$z_t = 1$	2	3
$z_{t-1} = 1$	0.5	0.5	0
2	0.1	0.8	0.1
3	0	0.5	0.5

$p(x_t\|z_t)$	$x_t = $ A	C	G	T
$z_t = 1$	0.3	0.1	0	0.6
2	0	0.8	0.2	0
3	0	0.1	0.9	0

$p(z_1, z_2\|x_1 = C, x_2 = C)$	$z_2 = 1$	2	3
$z_1 = 1$	0.008726	0.07	0.
2	0.014	0.89	0.014
3	0.	0.	0.

posterior over the latent sequence

$p(z_1, z_2)$	$z_2 = 1$	2	3
$z_1 = 1$	0.25	0.25	0.
2	0.05	0.4	0.05
3	0.	0.	0.

prior over the latent sequence

Outline

- Hidden Markov models
  - Numerical example of figuring out marginal of the observed sequence
  - Numerical example of figuring out the most probable sequence
- **Message-passing algorithm**
  - Factor graph review
  - Sum-product algorithms

Factor graph

- Why Factor graphs (FGs)? **Problem**: computing marginals and posteriors are tedious and expensive in a graphical model (such as Ex. 1 and Ex. 2). We need an algorithm to automate the local summations in analogous to backpropagation.

- Advantage over BNs and MRFs:
  - FG provides a simple and unified way to think about local computations (factors and messages).
  - FG makes it easy to design message-passing algorithms that efficiently exploit local factorization structure in a graph.

The sum product algorithm on factor graph

- Our first message-passing algorithm: **the sum-product algorithm**

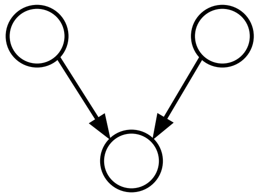
- The sum-product algorithm computes probabilities for a **subset** of the variables of a **factor graph** , e.g. P(a, b, c, d)
  - Marginal distributions , e.g. P(b)
  - Joint distributions of a subset of variables , e.g. P(a,b)
  - Conditional distributions (often the posterior distributions of our interest) , e.g. P(a,c | d)

= P(a,c,d) / P(d)

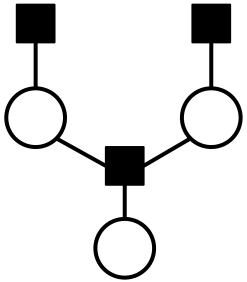
We need marginals to normalize the posterior distributions

Review: convert BNs, MRFs to factor graph

- Converting Bayesian Networks to factor graph takes the following steps:
    - Consider all the parents of a child node
    - “Pinch” all the edges from its parents to the child into one factor
    - Create an additional edge from the factor to the child node
    - Move on the the next child node
    - Last step is to add all the priors as individual “dongles” to the corresponding variables

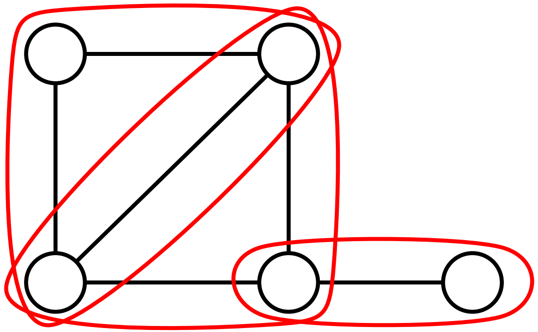


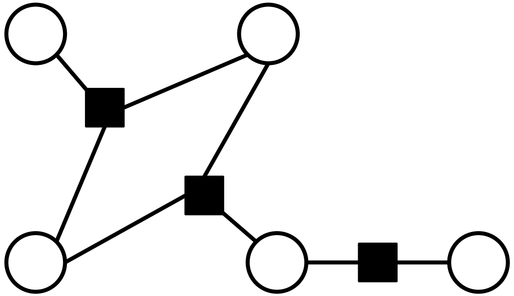
- **Sanity check**: Let the original BN have N variables and E edges.
  The converted factor graph will have N+E edges in total



Review: convert BNs, MRFs to factor graph

- Converting Markov Random Fields to factor graph takes the following steps:
  - Consider all the maximum cliques of the MRF
  - Create a factor node for each of the maximum cliques
  - Connect all the nodes of the maximum clique to the new factor nodes





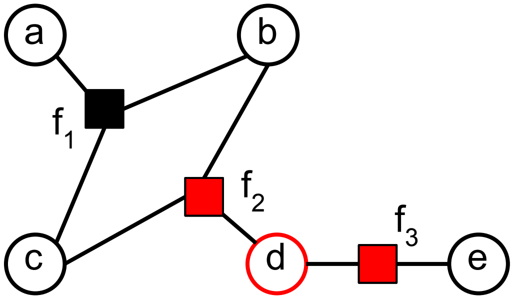
Review: conditional independence in factor graph

- The Markov blanket for our factor graphs is very similar to MRFs

- The Markov blanket of a variable node in a factor graph is given by the variables' **second-neighbour** nodes

Consider the nearest-neighbours of variable d:

$$Ne(d) = \{f_2, f_3\}$$



Review: conditional independence in factor graph

- The Markov blanket for our factor graphs is very similar to MRFs

- The Markov blanket of a variable node in a factor graph is given by the variables' **second-neighbour** nodes

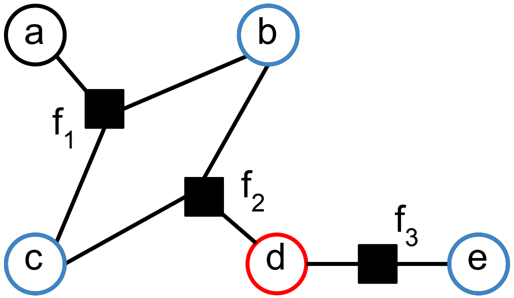
Set exclusion notation: $Ne(x) \setminus d$ denotes the neighbours of x with variable d excluded.

Consider the nearest-neighbours of variable d:

$$Ne(d) = \{f_2, f_3\}$$

Consider the second-neighbours of variable d:

$$\bigcup_{x \in Ne(d)} Ne(x) \backslash d = Ne(f_2) \setminus d \cup Ne(f_3) \setminus d = \{b, c, e\}$$

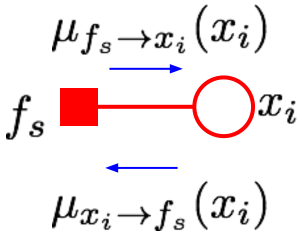


Message notations

- **On each edge** of the factor graph, there are **two messages** traveling in opposite directions.

- We use function μ to denote **messages**. The messages are scalar functions of the variable nodes.

- The subscript denotes the origin and the destination of these messages, e.g.:



factor-to-variable

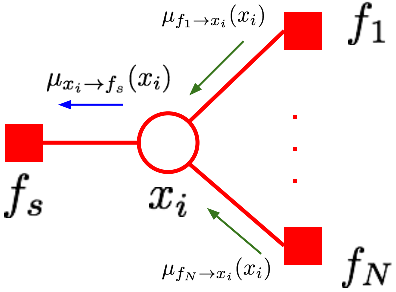
variable-to-factor

The sum product algorithm on factor graph

- Two rules in the sum-product algorithm:
  - Variable-to-factor messages:

$$\mu_{x_i \to f_s}(x_i) = \prod_{f_n \in Ne(x_i) \backslash f_s} \underbrace{\mu_{f_n \to x_i}(x_i)}_{\text{Incoming messages}}$$

Let $\{f_s, f_1, \dots, f_N\}$ be a subset of the factor nodes in a factor graph connected to variable xi, $Ne(x_i) = \{f_s, f_1, \dots, f_N\}$



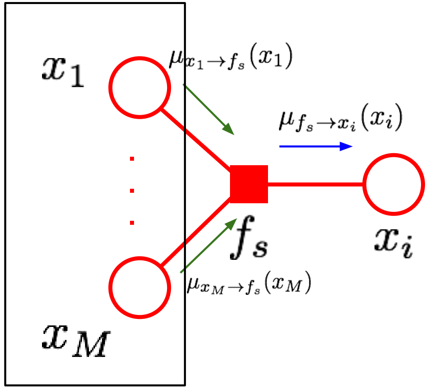
There are N factor nodes in total connected to xi: $|\{f_s, f_1, \dots, f_N)| = N$

The sum product algorithm on factor graph

- Two rules in the sum-product algorithm:
  - Factor-to-variable messages:

Let $\{x_i, x_1, \cdots, x_M\}$ be a subset of the variable nodes in a factor graph connected to factor fs.
$Ne(f_s) = \{x_i, x_1, \ldots, x_M\}$

$$\mu_{f_s \to x_i}(x_i) = \sum_{Ne(f_s) \backslash x_i} \left[f_s(x_i, x_1, \ldots, x_M) \underbrace{\prod_{x_m \in Ne(f_s) \backslash x_i} \mu_{x_m \to f_s}(x_m)}_{\text{Incoming messages}} \right]$$



Note that: $Ne(f_s) \setminus x_i = \{x_j : j \in \{1, \ldots, M\} \setminus i\}$

There are M variable nodes in total connected to fs: $|\{x_i, x_1, \ldots, x_M)| = M$

The sum product algorithm on factor graph

- Two rules in the sum-product algorithm:
  - Factor-to-variable messages:

$$\mu_{f_s \to x_i}(x_i) = \sum_{Ne(f_s) \backslash x_i} \left[f_s(x_i, x_1, \ldots, x_M) \prod_{x_m \in Ne(f_s) \backslash x_i} \mu_{x_m \to f_s}(x_m) \right]$$

  - Variable-to-factor messages:

$$\mu_{x_i \to f_s}(x_i) = \prod_{f_n \in Ne(x_i) \backslash f_s} \mu_{f_n \to x_i}(x_i)$$

The sum product algorithm on factor graph

- How to start the sum-product algorithm:
    - Choose a node in the factor graph as the root node
    - Compute all the leaf-to-root messages
    - Compute all the root-to-leaf messages

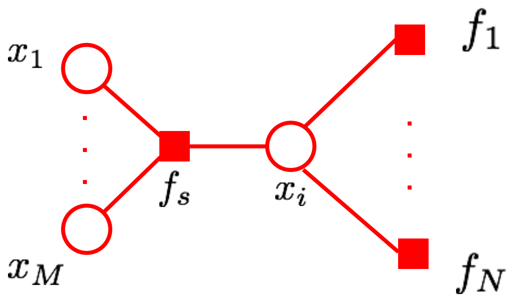
The sum product algorithm on factor graph

- How to start the sum-product algorithm:
  - Choose a node in the factor graph as the root node
  - Compute all the leaf-to-root messages
  - Compute all the root-to-leaf messages

- Initial conditions (i.e. the nearest-neighbours exclude the outgoing node is empty set $Ne(f_s) \setminus x_i = \emptyset$ or $Ne(x_i) \setminus f_s = \emptyset$):
  - Starting from a factor leaf/root node, the initial factor-to-variable message is the factor itself
  - Starting from a variable leaf/root node, the initial variable-to-factor message is a vector of ones

Compute probabilities with messages

- How to convert messages to actual probabilities:



Define **unnormalized probabilities** as g. That is, g is positive everywhere and may not sum to 1.

$$Z = \sum_{x_i, x_1, \cdots, x_M} g(x_i, x_1, \cdots, x_M)$$

We can normalize g to obtain the actual joint and marginal probabilities:

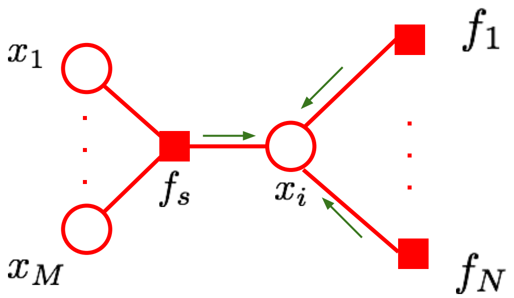
$$P(x_i, x_1, \cdots, x_M) = \frac{1}{Z} g(x_i, x_1, \cdots, x_M)$$

$$P(x_i) = \frac{1}{Z} g(x_i)$$

where Z is the normalization constant to get proper probability distributions

Compute probabilities with messages

- How to convert messages to actual probabilities:



Compute unnormalized probabilities using messages (the sum-product algorithm):

unnormalized marginal probabilities:

$$g(x_i) = \prod_{f_n \in Ne(x_i)} \mu_{f_n \to x_i}(x_i)$$

the product of all the incoming messages from the adjacent factors.

normalization constant:

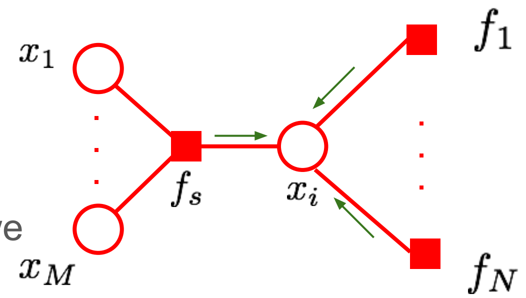
$$Z = \sum_{x_i} g(x_i)$$

marginal probabilities:

$$P(x_i) = \frac{1}{Z} g(x_i)$$

Compute probabilities with messages

- How to convert messages to actual probabilities:



Let $\{x_i\} \cup X_s \subseteq \{x_i, x_1, \cdots, x_M\}$ be a subset of the variables we wish to compute joint distribution over

Compute unnormalized probabilities using messages (the sum-product algorithm):

unnormalized joint probabilities:

$$g(x_i, X_s) = \prod_{f_n \in Ne(x_i)} \mu_{f_n \to x_i}(x_i, X_s)$$

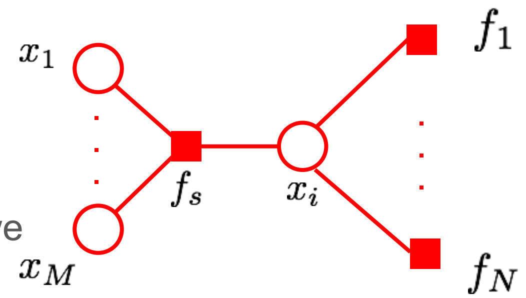
normalization constant:

$$Z = \sum_{x_i} g(x_i)$$

marginal probability of a subset of variables:

$$P(x_i, X_s) = \frac{1}{Z} g(x_i, X_s)$$

Compute probabilities with messages



- How to convert messages to actual probabilities:

Let $\{x_i\} \cup X_s \subseteq \{x_i, x_1, \cdots, x_M\}$ be a subset of the variables we wish to compute joint distribution over

Compute unnormalized probabilities using messages (the sum-product algorithm):

unnormalized joint probabilities:

$$g(x_i, X_s) = \prod_{f_n \in Ne(x_i)} \mu_{f_n \to x_i}(x_i, X_s)$$

e.g. When all the variables in $\{x_i\} \cup X_s$ are connected to the same factor. We make the computation efficient

$$g(x_i, X_s) = f_s(x_i, X_s) \prod_{x_m \in \{x_i, X_s\}} \prod_{f_n \in Ne(x_m) \setminus f_s} \mu_{f_n \to x_m}(x_m)$$

normalization constant:

$$Z = \sum_{x_i} g(x_i)$$

marginal probability of a subset of variables:

$$P(x_i, X_s) = \frac{1}{Z} g(x_i, X_s)$$