School of Computer Science
Carnegie Mellon

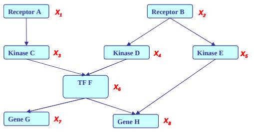
Junction Tree Algorithm

and a case study of the

Hidden Markov Model

Probabilistic Graphical Models (10-708)

Lecture 6, Oct 3, 2007



Eric Xing

Reading: J-Chap 12, 17, KF-Chap. 10

Outline

- So far we have studied exact inference in:
  - Trees
    - message passing on the original graph (which are trees)
  - Poly-trees, Tree-like graphs
    - message passing in factor trees
- Now we will look into exact inference in arbitrary graphs
  - Junction-Tree algorithm
- Inference in Hidden Markov Model

The Junction Tree Algorithm

- Recall: Elimination ≡ message passing on a **clique tree**
- Junction Tree Algorithm:
  - computing messages on a **clique tree**
  - message passing protocol on a **clique tree**
- There are several inference algorithms; some of which operate directly on (special) directed graph
  - Forward-backward algorithm for HMM (we will see it later)
  - Pealing algorithm for trees and phylogenies
- The junction tree algorithm is the most popular and general inference algorithm, it operates on an undirected graph
  - To understand the JT-algorithm, we need to understand how to compile a directed graph into an undirected graph

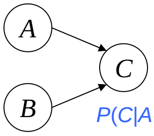
---

Moral Graph

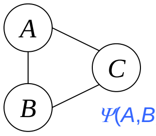
- Note that for both directed GMs and undirected GMs, the joint probability is in a product form:

BN: $P(\mathbf{X}) = \prod_{i=1:d} P(X_i \mid \mathbf{X}_{\pi_i})$ **MRF:** $P(\mathbf{X}) = \frac{1}{Z}\prod_{c \in C} \psi_c(\mathbf{X}_c)$

- So let's convert local conditional probabilities into potentials; then the second expression will be generic, but how does this operation affect the directed graph?
  - We can think of a conditional probability, e.g,. $P(C|A,B)$ as a function of the three variables A, B, and C (we get a real number of each configuration):



$P(C|A,B)$

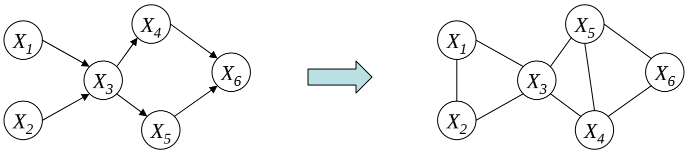


$\Psi(A,B,C) = P(C|A,B)$

  - Problem: But a node and its parent are not generally in the same clique in a BN
  - Solution: Marry the parents to obtain the "moral graph"

Moral Graph (cont.)

- Define the potential on a clique as the product over all conditional probabilities contained within the clique
- Now the product of potentials gives the right answer:



$$\begin{aligned} &P(X_1, X_2, X_3, X_4, X_5, X_6) \\ &= P(X_1)P(X_2)P(X_3 \mid X_1, X_2)P(X_4 \mid X_3)P(X_5 \mid X_3)P(X_6 \mid X_4, X_5) \\ &= \psi(X_1, X_2, X_3)\psi(X_3, X_4, X_5)\psi(X_4, X_5, X_6) \end{aligned}$$

where

$$\psi(X_1, X_2, X_3) = P(X_1)P(X_2)P(X_3 \mid X_1, X_2)$$

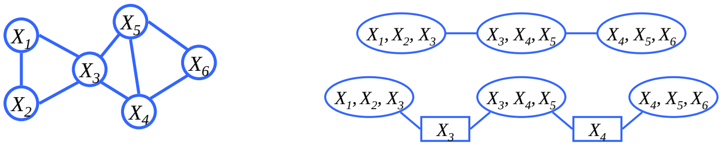
$$\psi(X_3, X_4, X_5) = P(X_4 \mid X_3)P(X_5 \mid X_3)$$

$$\psi(X_4, X_5, X_6) = P(X_6 \mid X_4, X_5)$$

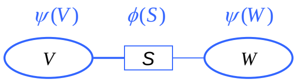
Note that here the interpretation of potential is ambivalent:
it can be either *marginals* or *conditionals*

Clique trees

- A clique tree is an (undirected) tree of cliques



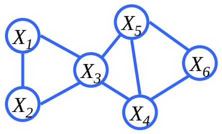
- Consider cases in which two neighboring cliques V and W have an overlap S (e.g., (X_1, X_2, X_3) overlaps with (X_3, X_4, X_5)),

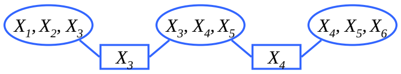


- Now we have an alternative representation of the joint in terms of the potentials:

Clique trees

- A clique tree is an (undirected) tree of cliques





- The alternative representation of the joint in terms of the potentials:

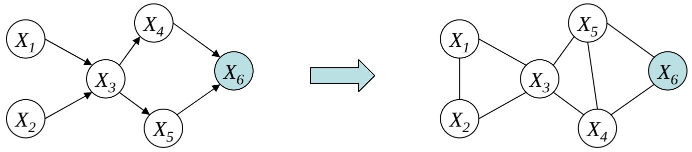
$$\begin{aligned}&P(X_1,X_2,X_3,X_4,X_5,X_6)\\ &=P(X_1)P(X_2)P(X_3\mid X_1,X_2)P(X_4\mid X_3)P(X_5\mid X_3)P(X_6\mid X_4,X_5)\\ &=P(X_1,X_2,X_3)\frac{P(X_3,X_4,X_5)}{P(X_3)}\frac{P(X_4,X_5,X_6)}{P(X_4,X_5)}\\ &=\psi(X_1,X_2,X_3)\frac{\psi(X_3,X_4,X_5)}{\phi(X_3)}\frac{\psi(X_4,X_5,X_6)}{\phi(X_4,X_5)}\end{aligned}$$

Now each potential is isomorphic to the *cluster marginal* of the attendant set of variables

- Generally: $\boxed{P(\mathbf{X})=\frac{\prod_C \psi_C(\mathbf{X}_C)}{\prod_S \phi_S(\mathbf{X}_S)}}$

Why this is useful?

- Propagation of probabilities
  - Now suppose that some evidence has been "absorbed" (i.e., certain values of some nodes have been observed). How do we propagate this effect to the rest of the graph?



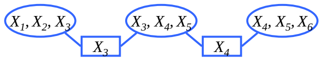
  - What do we mean by propagate?

    Can we adjust all the potentials $\{\psi\}$, $\{\phi\}$ so that they still represent the correct cluster marginals (or unnormalized equivalents) of their respective attendant variables?

  - Utility? $P(X_1 \mid X_6 = x_6) = \sum_{X_2,X_3} \psi(X_1,X_2,X_3)$

$$P(X_3 \mid X_6 = x_6) = \phi(X_3)$$

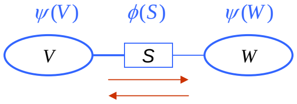
$$P(x_6) = \sum_{X_4,X_5} \psi(X_4,X_5,x_6)$$



Local operations!

Local Consistency

- We have two ways of obtaining $p(S)$



$$P(S) = \sum_{V\setminus S}\psi(V) \qquad P(S) = \sum_{W\setminus S}\psi(W)$$

and they must be the same

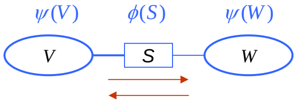
- The following update-rule ensures this:
  - Forward update: $\phi_S^* = \sum_{V\setminus S}\psi_V \quad \psi_W^* = \frac{\phi_S^*}{\phi_S}\psi_W$
  - Backward update $\phi_S^{**} = \sum_{W\setminus S}\psi_W^* \quad \psi_V^{**} = \frac{\phi_S^{**}}{\phi_S^*}\psi_V^*$
- Two important identities can be proven

$$\sum_{V\setminus S}\psi_V^{**} = \sum_{W\setminus S}\psi_W^* = \phi_S^{**} \qquad \frac{\psi_V^*\psi_W^*}{\phi_S^*} = \frac{\psi_V^{**}\psi_W^{**}}{\phi_S^{**}} = \frac{\psi_V\psi_W}{\phi_S}$$

Local Consistency

Invariant Joint

Message Passing Algorithm



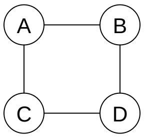
$$\phi_S^* = \sum_{V\setminus S}\psi_V \quad \psi_W^* = \frac{\phi_S^*}{\phi_S}\psi_W$$

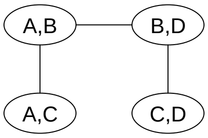
$$\phi_S^{**} = \sum_{W\setminus S}\psi_W^* \quad \psi_V^{**} = \frac{\phi_S^{**}}{\phi_S^*}\psi_V^*$$

- This simple local message-passing algorithm on a clique tree defines the general probability propagation algorithm for directed graphs!
  - Many interesting algorithms are special cases:
    - Forward-backward algorithm for hidden Markov models,
    - Kalman filter updates
    - Pealing algorithms for probabilistic trees
  - The algorithm seems reasonable. Is it correct?

A problem

- Consider the following graph and a corresponding clique tree

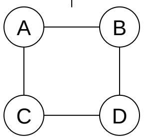




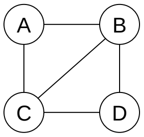
  - Note that C appears in two non-neighboring cliques
- *Question*: with the previous message passage, can we ensure that the probability associated with C in these two (non-neighboring) cliques consistent?
- Answer: No. It is not true that in general local consistency implies global consistency
- What else do we need to get such a guarantee?

Triangulation

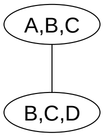
- A triangulated graph is one in which *no cycles* with four or more nodes exist in which there is no *chord*



- We triangulate a graph by adding chords:

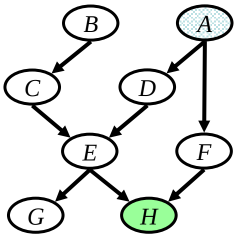


- Now we no longer have our global inconsistency problem.
  - A clique tree for a triangulated graph has the *running intersection property*: If a node appears in two cliques, it appears everywhere on the path between the cliques
  - Thus local consistency implies global consistency



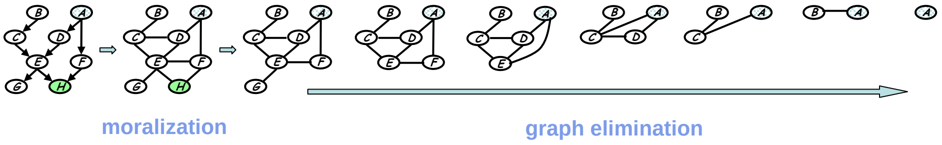
Junction trees

- A clique tree for a triangulated graph is referred to as a *junction tree*
- In junction trees, local consistency implies global consistency. Thus the local message-passing algorithms is (provably) correct
- It is also possible to show that *only* triangulated graphs have the property that their clique trees are junctions. Thus if we want local algorithms, we *must* triangulate
- Are we now all set?
  - How to triangulate?
  - The complexity of building a JT depends on how we triangulate!!
  - Consider this network:
    it turns out that we will need to pay an $O(2^4)$ or $O(2^6)$ cost depending on how we triangulate!



How to triangulate

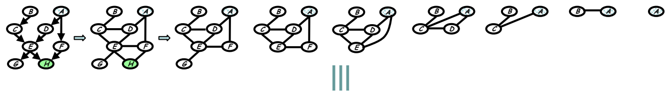
- A graph elimination algorithm

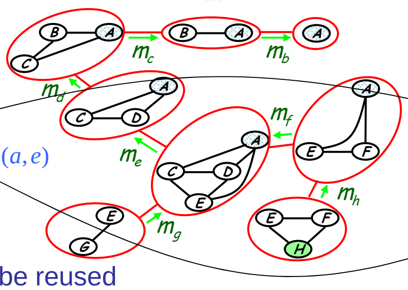


- Intermediate terms correspond to the cliques resulted from elimination
  - "good" elimination orderings lead to **small cliques** and hence reduce complexity (what will happen if we eliminate "e" first in the above graph?)
  - finding the optimum ordering is NP-hard, but for many graph optimum or near-optimum can often be heuristically found

From Elimination to Message Passing

- Our algorithm so far answers only one query (e.g., on one node), do we need to do a complete elimination for every such query?
- Elimination ≡ message passing on a **clique tree**





$$m_e(a,c,d) = \sum_e p(e|c,d) m_g(e) m_f(a,e)$$

Recall this:

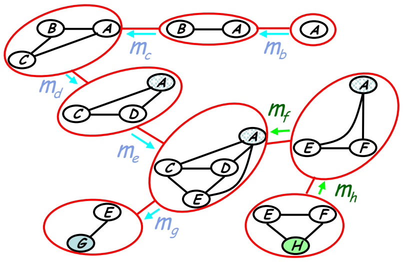
$$\phi_S^* = \sum_{V\setminus S} \psi_V$$

$$\psi_W^* = \frac{\phi_S^*}{\phi_S} \psi_W$$

- Messages can be reused

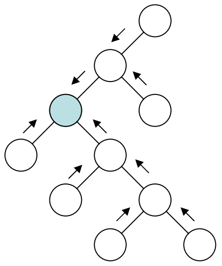
From Elimination to Message Passing

- Our algorithm so far answers only one query (e.g., on one node), do we need to do a complete elimination for every such query?
- Elimination ≡ message passing on a **clique tree**
  - **Another query ...**

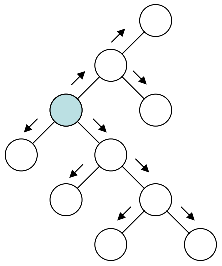


- Messages m_f and m_h are reused, others need to be recomputed

Message-passing algorithms



collect



distribute

- Message update
  - The Hugin update
  - The Shafer-Shenoy update

$$\phi_S^* = \sum_{V\setminus S}\psi_V \qquad \psi_W^* = \frac{\phi_S^*}{\phi_S}\psi_W$$

$$m_{i\to j}(S_{ij}) = \sum_{C_i\setminus S_{ij}} \psi_{C_i} \prod_{k\neq j} m_{k\to i}(S_{ki})$$

A Sketch of the Junction Tree Algorithm

- **The algorithm**

1. Moralize the graph (trivial)
2. Triangulate the graph (good heuristic exist, but actually NP hard)
3. Build a clique tree (e.g., using a maximum spanning tree algorithm
4. Propagation of probabilities --- a local message-passing protocol

- Results in marginal probabilities of all cliques --- solves all queries in a single run
- A **generic** exact inference algorithm for any GM
- **Complexity**: exponential in the size of the maximal clique --- a good elimination order often leads to small maximal clique, and hence a good (i.e., thin) JT

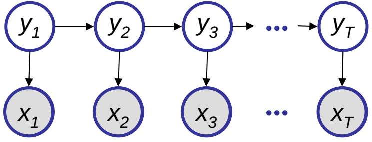
Summary

- Junction tree data-structure for exact inference on general graphs
- Two methods
  - Shafer-Shenoy
  - Belief-update or Lauritzen-Speigelhalter
- Constructing Junction tree from chordal graphs
  - Maximum spanning tree approach

Case study:

- **Hidden Markov Model**

Recall definition of HMM



- Transition probabilities between any two states

$$p(y_t^j = 1 \mid y_{t-1}^i = 1) = a_{i,j},$$

or $p(y_t \mid y_{t-1}^i = 1) \sim \text{Multinomial}\left(a_{i,1}, a_{i,2}, \ldots, a_{i,M}\right), \forall i \in \mathbb{I}.$

- Start probabilities

$$p(y_1) \sim \text{Multinomial}\left(\pi_1, \pi_2, \ldots, \pi_M\right).$$

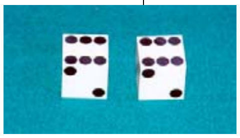
- Emission probabilities associated with each state

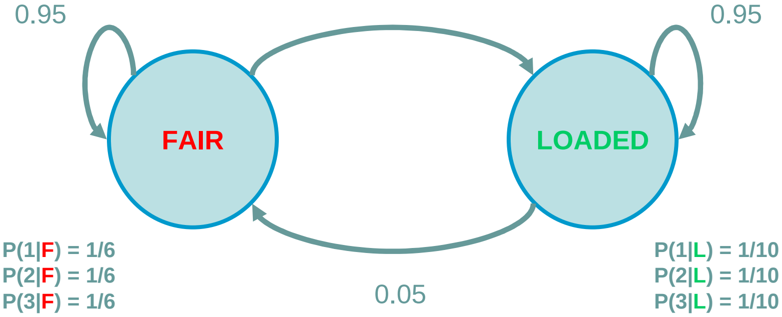
$$p(x_t \mid y_t^i = 1) \sim \text{Multinomial}\left(b_{i,1}, b_{i,2}, \ldots, b_{i,K}\right), \forall i \in \mathbb{I}.$$

or in general: $p(x_t \mid y_t^i = 1) \sim \mathrm{f}\left(\cdot \mid \theta_i\right), \forall i \in \mathbb{I}.$

The Dishonest Casino Model

Transition probabilities: $p(y_t^j = 1 \mid y_{t-1}^i = 1) = a_{i,j},$

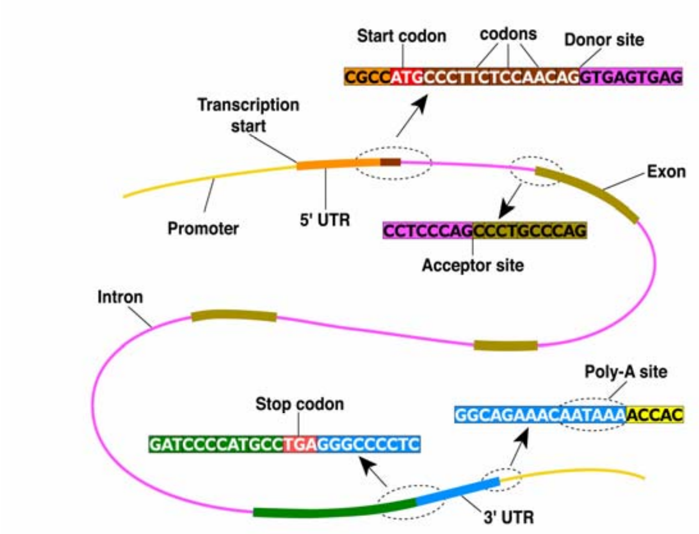




P(1|**F**) = 1/6
P(2|**F**) = 1/6
P(3|**F**) = 1/6
P(4|**F**) = 1/6
P(5|**F**) = 1/6
P(6|**F**) = 1/6

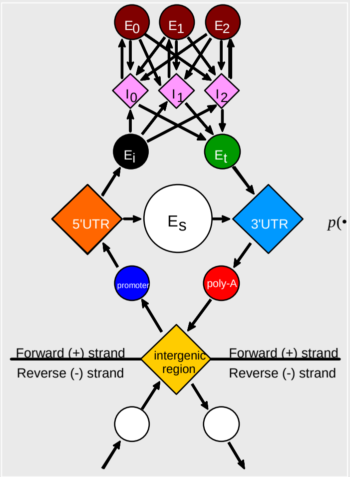
P(1|**L**) = 1/10
P(2|**L**) = 1/10
P(3|**L**) = 1/10
P(4|**L**) = 1/10
P(5|**L**) = 1/10
P(6|**L**) = 1/2

Typical structure of a gene

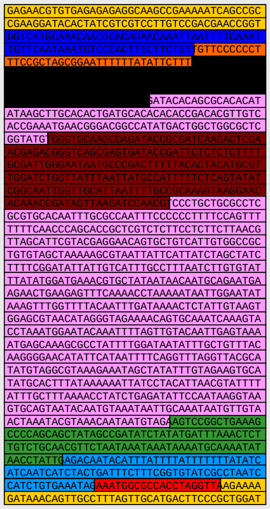


GENSCAN (Burge & Karlin)

Transition probabilities: $p(y_t^j = 1 \mid y_{t-1}^i = 1) = a_{i,j}$,



$$p(\bullet \mid y) = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix}$$

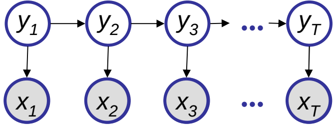


```
GAGAACGTGTGAGAGAGAGGCAAGCCGAAAAATCAGCCGC
CGAAGGATACACTATCGTCGTCCTTGTCCGACGAACCGGT
GGTCATGCAAACAACGCACAGAACAAATTAATTTTCAAAT
TGTTCAATAAATGTCCCACTTGCTTCTGTTGTTCCCCCCT
TTCCGCTAGCGGAATTTTTATATTCTTT


GATACACAGCGCACACAT
ATAAGCTTGCACACTGATGCACACACACCGACACGTTGTC
ACCGAAATGAACGGGACGGCCATATGACTGGCTGGCGCTC
GGTATGTGGGTGCAAGCGAGATACCGCGATCAAGACTCGA
ACGAGACGGGTCAGCGAGTGATACCGATTCTCTCTCTTTT
GCGATTGGGAATAATGCCCGACTTTTTACACTACATGCGT
TGGATCTGGTTATTTAATTATGCCATTTTTCTCAGTATAT
CGGCAATTGGTTGCATTAATTTTGCCGCAAAGTAAGGAAC
ACAAACCGATAGTTAAGATCCAACGTCCCTGCTGCGCCTC
GCGTGCACAATTTGCGCCAATTTCCCCCCTTTTCCAGTTT
TTTTCAACCCAGCACCGCTCGTCTCTTCCTCTTCTTAACG
TTAGCATTCGTACGAGGAACAGTGCTGTCATTGTGGCCGC
TGTGTAGCTAAAAAGCGTAATTATTCATTATCTAGCTATC
TTTTCGGATATTATTGTCATTTGCCTTTAATCTTGTGTAT
TTATATGGATGAAACGTGCTATAATAACAATGCAGAATGA
AGAACTGAAGAGTTTCAAAACCTAAAAATAATTGGAATAT
AAAGTTTGGTTTTACAATTTGATAAAACTCTATTGTAAGT
GGAGCGTAACATAGGGTAGAAAACAGTGCAAATCAAAGTA
CCTAAATGGAATACAAATTTTAGTTGTACAATTGAGTAAA
ATGAGCAAAGCGCCTATTTTGGATAATATTTGCTGTTTAC
AAGGGGAACATATTCATAATTTTCAGGTTTAGGTTACGCA
TATGTAGGCGTAAAGAAATAGCTATATTTGTAGAAGTGCA
TATGCACTTTATAAAAAATTATCCTACATTAACGTATTTT
ATTTGCTTTAAAACCTATCTGAGATATTCCAATAAGGTAA
GTGCAGTAATACAATGTAAATAATTGCAAATAATGTTGTA
ACTAAATACGTAAACAATAATGTAGAAGTCCGGCTGAAAG
CCCCAGCAGCTATAGCCGATATCTATATGATTTAAACTCT
TGTCTGCAACGTTCTAATAAATAAATAAAATGCAAAATAT
AACCTATTGAGACAATACATTTATTTTATTTTTTTATATC
ATCAATCATCTACTGATTTCTTTCGGTGTATCGCCTAATC
CATCTGTGAAATAGAAATGGCGCCACCTAGGTTAAGAAAA
GATAAACAGTTGCCTTTAGTTGCATGACTTCCCGCTGGAT
```

Probability of a parse

- Given a sequence $\mathbf{x} = x_1\ldots\ldots x_T$ and a parse $\mathbf{y} = y_1, \ldots\ldots, y_T$,
- To find how likely is the parse: (given our HMM and the sequence)



$$p(\mathbf{x}, \mathbf{y}) = p(x_1\ldots\ldots x_T, y_1, \ldots\ldots, y_T) \qquad \text{(Joint probability)}$$

$$= p(y_1)\, p(x_1 \mid y_1)\, p(y_2 \mid y_1)\, p(x_2 \mid y_2) \ldots p(y_T \mid y_{T-1})\, p(x_T \mid y_T)$$

$$= p(y_1, \ldots\ldots, y_T)\, p(x_1\ldots\ldots x_T \mid y_1, \ldots\ldots, y_T)$$

Let $\pi_{y_1} \overset{\text{def}}{=} \prod_{i=1}^{M} [\pi_i]^{y_1^i}$, $\quad a_{y_t, y_{t+1}} \overset{\text{def}}{=} \prod_{i,j=1}^{M} [a_{ij}]^{y_t^i y_{t+1}^j}$, $\quad$ and $\; b_{y_t, x_t} \overset{\text{def}}{=} \prod_{i=1}^{M} \prod_{k=1}^{K} [b_{ik}]^{y_t^i x_t^k}$,

$$= \pi_{y_1} a_{y_1, y_2} \cdots a_{y_{T-1}, y_T} \quad b_{y_1, x_1} \cdots b_{y_T, x_T}$$

- Marginal probability: $p(\mathbf{x}) = \sum_{\mathbf{y}} p(\mathbf{x}, \mathbf{y}) = \sum_{y_1} \sum_{y_2} \cdots \sum_{y_N} \pi_{y_1} \prod_{t=2}^{T} a_{y_{t-1}, y_t} \prod_{t=1}^{T} p(x_t \mid y_t)$
- Posterior probability: $p(\mathbf{y} \mid \mathbf{x}) = p(\mathbf{x}, \mathbf{y}) / p(\mathbf{x})$

Applications of HMMs

- **Some early applications of HMMs**
  - finance, but we never saw them
  - speech recognition
  - modelling ion channels
- In the mid-late 1980s HMMs entered genetics and molecular biology, and they are now firmly entrenched.
- **Some current applications of HMMs to biology**
  - mapping chromosomes
  - aligning biological sequences
  - predicting sequence structure
  - inferring evolutionary relationships
  - finding genes in DNA sequence

Three main questions on HMMs

1. Evaluation

GIVEN	an HMM ***M***, and a sequence $\mathbf{x}$,
FIND	Prob ($\mathbf{x}$ | ***M***)
ALGO.	Forward

2. Decoding

GIVEN	an HMM ***M***, and a sequence $\mathbf{x}$,
FIND	the sequence $\mathbf{y}$ of states that maximizes, e.g., P($\mathbf{y}$ | $\mathbf{x}$, ***M***), or the most probable subsequence of states
ALGO.	Viterbi, Forward-backward

3. Learning (next lecture)

GIVEN	an HMM ***M***, with unspecified transition/emission probs., and a sequence $\mathbf{x}$,
FIND	parameters $\theta = (\pi_i, a_{ij}, \eta_{ik})$ that maximize P($\mathbf{x}$ | θ)
ALGO.	Baum-Welch (EM)

The Forward Algorithm

- We want to calculate $P(\mathbf{x})$, the likelihood of $\mathbf{x}$, given the HMM
  - Sum over all possible ways of generating $\mathbf{x}$:

$$p(\mathbf{x}) = \sum_{\mathbf{y}} p(\mathbf{x},\mathbf{y}) = \sum_{y_1}\sum_{y_2}\cdots\sum_{y_N} \pi_{y_1}\prod_{t=2}^{T} a_{y_{t-1},y_t}\prod_{t=1}^{T} p(x_t \mid y_t)$$

  - To avoid summing over an exponential number of paths $\mathbf{y}$, define

$$\alpha(y_t^k = 1) = \alpha_t^k \overset{\text{def}}{=} P(x_1,\ldots,x_t, y_t^k = 1)$$

(the forward probability)

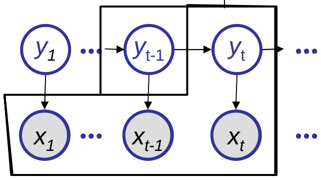
  - The recursion:

$$\alpha_t^k = p(x_t \mid y_t^k = 1)\sum_i \alpha_{t-1}^i a_{i,k}$$

$$P(\mathbf{x}) = \sum_k \alpha_T^k$$

The Forward Algorithm – derivation

- Compute the forward probability:



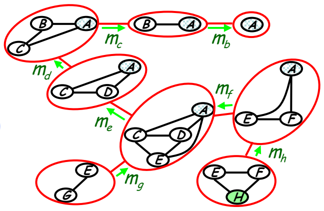
$$\alpha_t^k = P(x_1,\ldots,x_{t-1},x_t,y_t^k=1)$$
$$=\sum_{y_{t-1}} P(x_1,\ldots,x_{t-1},x_t,y_{t-1},y_t^k=1)$$
$$=\sum_{y_{t-1}} P(x_1,\ldots,x_{t-1},y_{t-1})P(y_t^k=1\mid y_{t-1},x_1,\ldots,x_{t-1})P(x_t\mid y_t^k=1,x_1,\ldots,x_{t-1},y_{t-1})$$
$$=\sum_{y_{t-1}} P(x_1,\ldots,x_{t-1},y_{t-1})P(y_t^k=1\mid y_{t-1})P(x_t\mid y_t^k=1)$$
$$=P(x_t\mid y_t^k=1)\sum_i P(x_1,\ldots,x_{t-1},y_{t-1}^i=1)P(y_t^k=1\mid y_{t-1}^i=1)$$
$$=P(x_t\mid y_t^k=1)\sum_i \alpha_{t-1}^i a_{i,k}$$

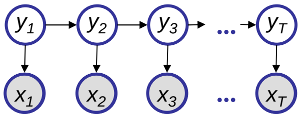
Chain rule: $P(A,B,C)=P(A)P(B\mid C)P(C\mid A,B)$

Recall the Elimination and Message Passing Algorithm

- Elimination $\equiv$ message passing on a **clique tree**

$$m_e(a,c,d) = \sum_e p(e\mid c,d)m_g(e)m_f(a,e)$$





$$\alpha_t^k = p(x_t\mid y_t^k=1)\sum_i \alpha_{t-1}^i a_{i,k}$$

$$P(\mathbf{x}) = \sum_k \alpha_T^k$$

The Forward Algorithm

- We can compute α_t^k for all k, t, using dynamic programming!

Initialization:

$$\alpha_1^k = P(x_1 \mid y_1^k = 1)\pi_k$$

$$\begin{aligned}\alpha_1^k &= P(x_1, y_1^k = 1) \\ &= P(x_1 \mid y_1^k = 1)P(y_1^k = 1) \\ &= P(x_1 \mid y_1^k = 1)\pi_k\end{aligned}$$

Iteration:

$$\alpha_t^k = P(x_t \mid y_t^k = 1)\sum_i \alpha_{t-1}^i a_{i,k}$$

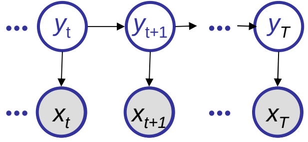
Termination:

$$P(\mathbf{x}) = \sum_k \alpha_T^k$$

The Backward Algorithm

- We want to compute $P(y_t^k = 1 \mid \mathbf{x})$,

  the posterior probability distribution on the t th position, given $\mathbf{x}$



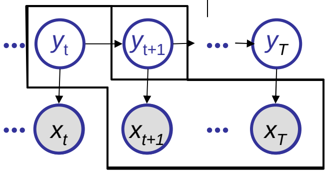
  - We start by computing

$$\begin{aligned}P(y_t^k = 1, \mathbf{x}) &= P(x_1, \dots, x_t, y_t^k = 1, x_{t+1}, \dots, x_T) \\ &= P(x_1, \dots, x_t, y_t^k = 1)P(x_{t+1}, \dots, x_T \mid x_1, \dots, x_t, y_t^k = 1) \\ &= \underbrace{P(x_1 \dots x_t, y_t^k = 1)}_{\text{Forward, } \alpha_t^k}\underbrace{P(x_{t+1} \dots x_T \mid y_t^k = 1)}_{\text{Backward, } \beta_t^k = P(x_{t+1}, \dots, x_T \mid y_t^k = 1)}\end{aligned}$$

  - The recursion: $\beta_t^k = \sum_i a_{k,i}\, p(x_{t+1} \mid y_{t+1}^i = 1)\beta_{t+1}^i$

The Backward Algorithm – derivation

- Define the backward probability:



$$\beta_t^k = P(x_{t+1},\dots,x_T \mid y_t^k = 1)$$
$$= \sum_{y_{t+1}} P(x_{t+1},\dots,x_T, y_{t+1} \mid y_t^k = 1)$$
$$= \sum_i P(y_{t+1}^i = 1 \mid y_t^k = 1) p(x_{t+1} \mid y_{t+1}^i = 1, y_t^k = 1) P(x_{t+2},\dots,x_T \mid x_{t+1}, y_{t+1}^i = 1, y_t^k = 1)$$
$$= \sum_i P(y_{t+1}^i = 1 \mid y_t^k = 1) p(x_{t+1} \mid y_{t+1}^i = 1) P(x_{t+2},\dots,x_T \mid y_{t+1}^i = 1)$$
$$= \sum_i a_{k,i} p(x_{t+1} \mid y_{t+1}^i = 1) \beta_{t+1}^i$$

$$\text{Chain rule}: \; P(A,B,C \mid \alpha) = P(A,\alpha)P(B \mid C,\alpha)P(C \mid A,B,\alpha)$$

The Backward Algorithm

- We can compute β_t^k for all k, t, using dynamic programming!

Initialization:

$$\beta_T^k = 1, \ \forall k$$

Iteration:

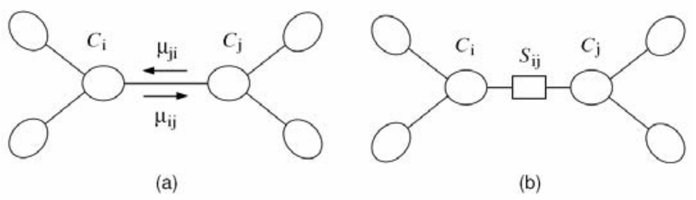
$$\beta_t^k = \sum_i a_{k,i} P(x_{t+1} \mid y_{t+1}^i = 1) \beta_{t+1}^i$$

Termination:

$$P(\mathbf{x}) = \sum_k \alpha_1^k \beta_1^k$$

Shafer Shenoy for HMMs

- Recap: Shafer-Shenoy algorithm



  - Message from clique *i* to clique *j* :

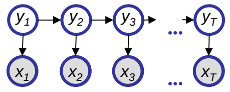
$$\mu_{i\to j}=\sum_{C_i\setminus S_{ij}}\psi_{C_i}\prod_{k\neq j}\mu_{k\to i}(S_{ki})$$

  - Clique marginal

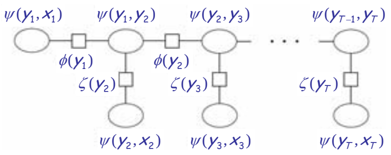
$$p(C_i)\propto\psi_{C_i}\prod_{k}\mu_{k\to i}(S_{ki})$$

Message Passing for HMMs (cont.)

- A junction tree for the HMM

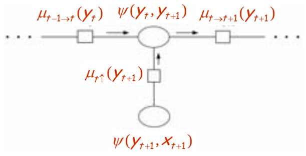


$\Rightarrow$



- Rightward pass

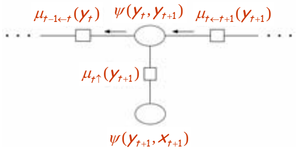
$$\mu_{t\to t+1}(y_{t+1})=\sum_{y_t}\psi(y_t,y_{t+1})\mu_{t-1\to t}(y_t)\mu_{t\uparrow}(y_{t+1})$$
$$=\sum_{y_t}p(y_{t+1}\mid y_t)\mu_{t-1\to t}(y_t)p(x_{t+1}\mid y_{t+1})$$
$$=p(x_{t+1}\mid y_{t+1})\sum_{y_t}a_{y_t,y_{t+1}}\mu_{t-1\to t}(y_t)$$



  - This is exactly the *forward algorithm*!

- Leftward pass …

$$\mu_{t-1\leftarrow t}(y_t)=\sum_{y_{t+1}}\psi(y_t,y_{t+1})\mu_{t\leftarrow t+1}(y_{t+1})\mu_{t\uparrow}(y_{t+1})$$
$$=\sum_{y_{t+1}}p(y_{t+1}\mid y_t)\mu_{t\leftarrow t+1}(y_{t+1})p(x_{t+1}\mid y_{t+1})$$



  - This is exactly the *backward algorithm*!

Posterior decoding

- We can now calculate

$$P(y_t^k = 1 | \mathbf{x}) = \frac{P(y_t^k = 1, \mathbf{x})}{P(\mathbf{x})} = \frac{\alpha_t^k \beta_t^k}{P(\mathbf{x})}$$

- Then, we can ask
  - What is the most likely state at position *t* of sequence $\mathbf{x}$:

$$k_t^* = \arg\max_k P(y_t^k = 1 | \mathbf{x})$$

  - Note that this is an MPA of a single hidden state, what if we want to a MPA of a whole hidden state sequence?
  - Posterior Decoding: $\left\{ y_t^{k_t^*} = 1 : t = 1 \cdots T \right\}$
  - This is different from MPA of a whole sequence of hidden states
  - This can be understood as *bit error rate* vs. *word error rate*

Example:
MPA of *X* ?
MPA of *(X, Y)* ?

x	*y*	*P(x,y)*
0	*0*	*0.35*
0	*1*	*0.05*
1	*0*	*0.3*
1	*1*	*0.3*

Viterbi decoding

- GIVEN $\mathbf{x} = x_1, \ldots, x_T$, we want to find $\mathbf{y} = y_1, \ldots, y_T$, such that $P(\mathbf{y}|\mathbf{x})$ is maximized:

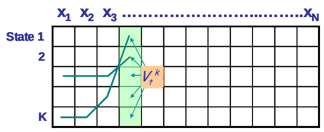
$$\mathbf{y}^* = \text{argmax}_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \text{argmax}_{\pi} P(\mathbf{y},\mathbf{x})$$

  - Let

$$V_t^k = \max_{\{y_1,\ldots y_{t-1}\}} P(x_1,\ldots,x_{t-1},y_1,\ldots,y_{t-1},x_t,y_t^k = 1)$$

= Probability of most likely **sequence of states** ending at state $y_t = k$

  - The recursion:



$$V_t^k = p(x_t | y_t^k = 1) \max_i a_{i,k} V_{t-1}^i$$

  - Underflows are a significant problem

$$p(x_1,\ldots,x_t,y_1,\ldots,y_t) = \pi_{y_1} a_{y_1,y_2} \cdots a_{y_{t-1},y_t} b_{y_1,x_1} \cdots b_{y_t,x_t}$$

    - These numbers become extremely small – underflow
    - Solution: Take the logs of all values: $V_t^k = \log p(x_t | y_t^k = 1) + \max_i \left(\log(a_{i,k}) + V_{t-1}^i\right)$

The Viterbi Algorithm – derivation

- Define the viterbi probability:

$$
\begin{aligned}
V_{t+1}^{k} &= \max_{\{y_1,\dots y_t\}} P(x_1,\dots,x_t,y_1,\dots,y_t,x_{t+1,}y_{t+1}^{k}=1)\\
&= \max_{\{y_1,\dots y_t\}} P(x_{t+1,}y_{t+1}^{k}=1 \mid x_1,\dots,x_t,y_1,\dots,y_t)P(x_1,\dots,x_t,y_1,\dots,y_t)\\
&= \max_{\{y_1,\dots y_t\}} P(x_{t+1,}y_{t+1}^{k}=1 \mid y_t)P(x_1,\dots,x_{t-1},y_1,\dots,y_{t-1},x_t,y_t)\\
&= \max_i P(x_{t+1,}y_{t+1}^{k}=1 \mid y_t^i=1)\max_{\{y_1,\dots y_{t-1}\}} P(x_1,\dots,x_{t-1},y_1,\dots,y_{t-1},x_t,y_t^i=1)\\
&= \max_i P(x_{t+1,} \mid y_{t+1}^{k}=1)a_{i,k}V_t^i\\
&= P(x_{t+1,} \mid y_{t+1}^{k}=1)\max_i a_{i,k}V_t^i
\end{aligned}
$$

The Viterbi Algorithm

- Input: $\mathbf{x} = x_1, \dots, x_T,$

Initialization:

$$V_1^k = P(x_1 \mid y_1^k = 1)\pi_k$$

Iteration:

$$V_t^k = P(x_{t,} \mid y_t^k = 1)\max_i a_{i,k}V_{t-1}^i$$

$$\mathrm{Ptr}(k,t) = \arg\max_i a_{i,k}V_{t-1}^i$$

Termination:

$$P(\mathbf{x},\mathbf{y}^*) = \max_k V_T^k$$

TraceBack:

$$y_T^* = \arg\max_k V_T^k$$

$$y_{t-1}^* = \mathrm{Ptr}(y_t^*, t)$$

Computational Complexity and implementation details

- What is the running time, and space required, for Forward, and Backward?

$$\alpha_t^k = p(x_t \mid y_t^k = 1) \sum_i \alpha_{t-1}^i a_{i,k}$$

$$\beta_t^k = \sum_i a_{k,i}\, p(x_{t+1} \mid y_{t+1}^i = 1) \beta_{t+1}^i$$

$$V_t^k = p(x_t \mid y_t^k = 1) \max_i a_{i,k} V_{t-1}^i$$

Time: $O(K^2N)$; Space: $O(KN)$.

- Useful implementation technique to avoid underflows
  - Viterbi: sum of logs
  - Forward/Backward: rescaling at each position by multiplying by a constant