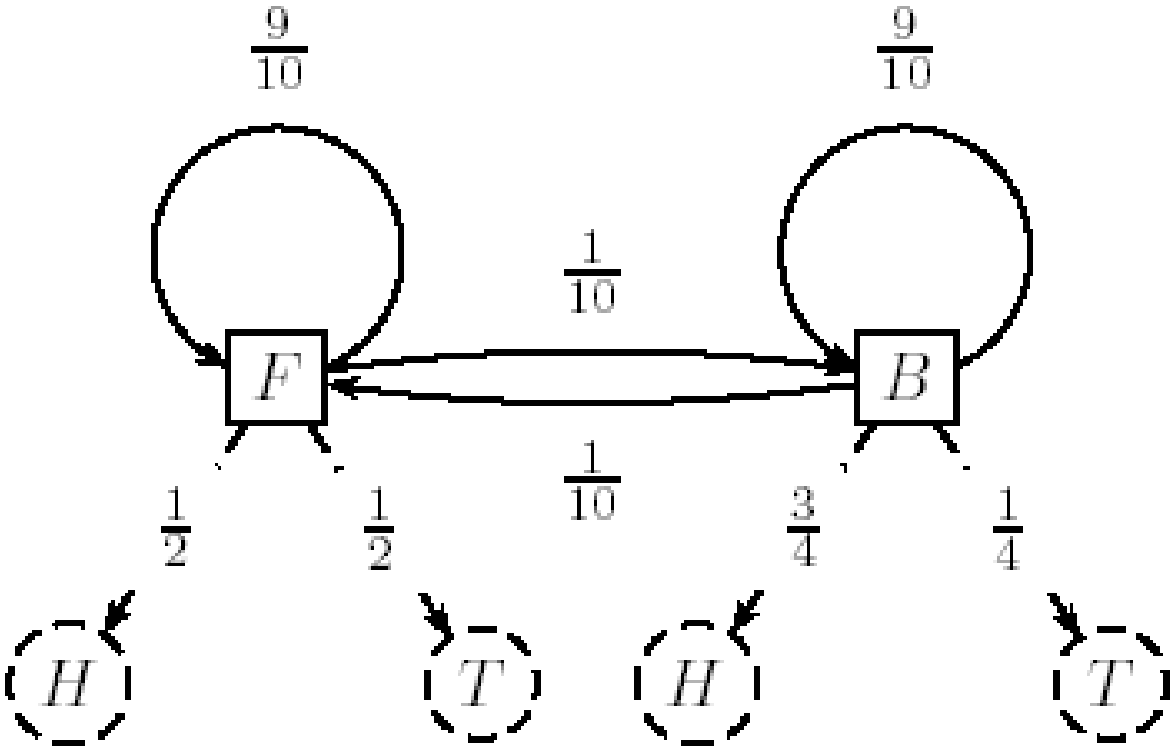
CS481: Bioinformatics Algorithms

Can Alkan
EA224
calkan@cs.bilkent.edu.tr

http://www.cs.bilkent.edu.tr/~calkan/teaching/cs481/

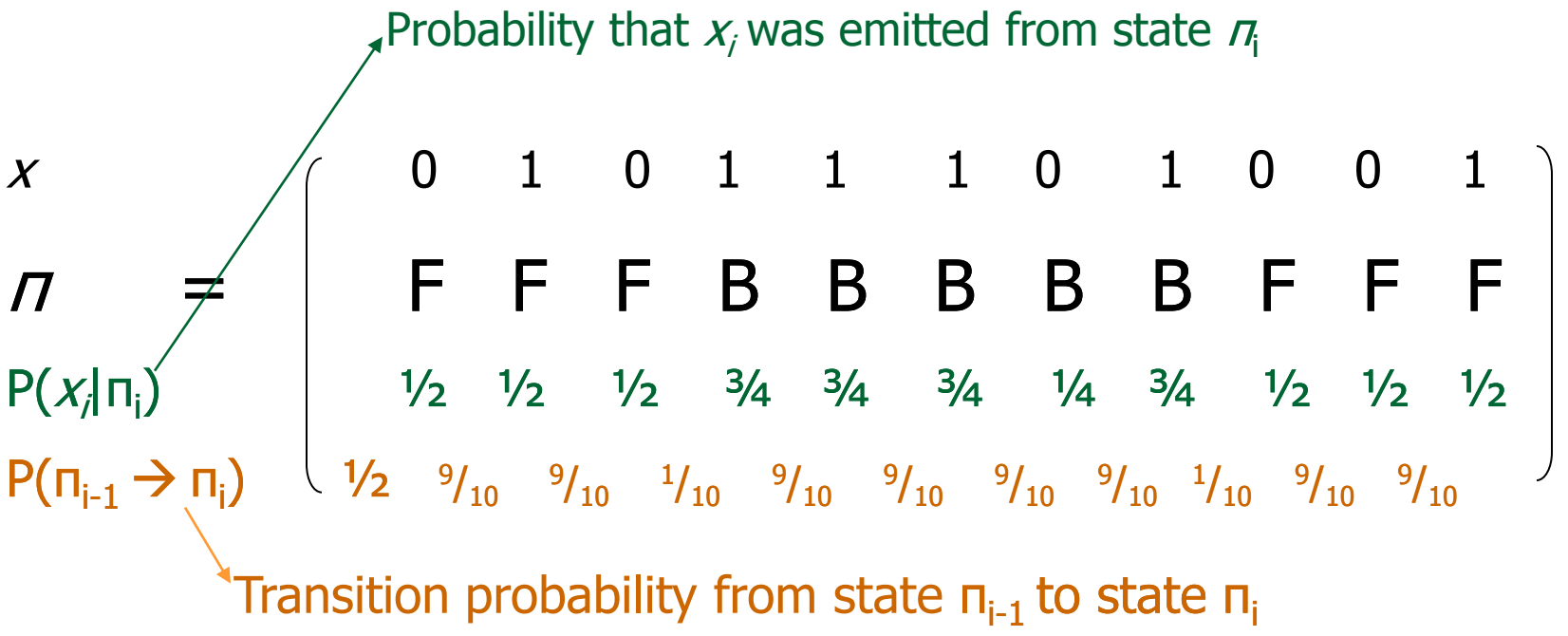
HMM for Fair Bet Casino (cont'd)



HMM model for the *Fair Bet Casino* Problem

Hidden Paths

- A *path* $\pi = \pi_1 \ldots \pi_n$ in the HMM is defined as a sequence of states.
- Consider path π = FFFBBBBBFFF and sequence x = 01011101001



P($x|\pi$) Calculation

- P($x|\pi$): Probability that sequence x was generated by the path π:

$$P(x|\pi) = P(\pi_0 \rightarrow \pi_1) \cdot \prod_{i=1}^{n} P(x_i|\pi_i) \cdot P(\pi_i \rightarrow \pi_{i+1})$$

$$= a_{\pi_0, \pi_1} \cdot \prod e_{\pi_i}(x_i) \cdot a_{\pi_i, \pi_{i+1}}$$

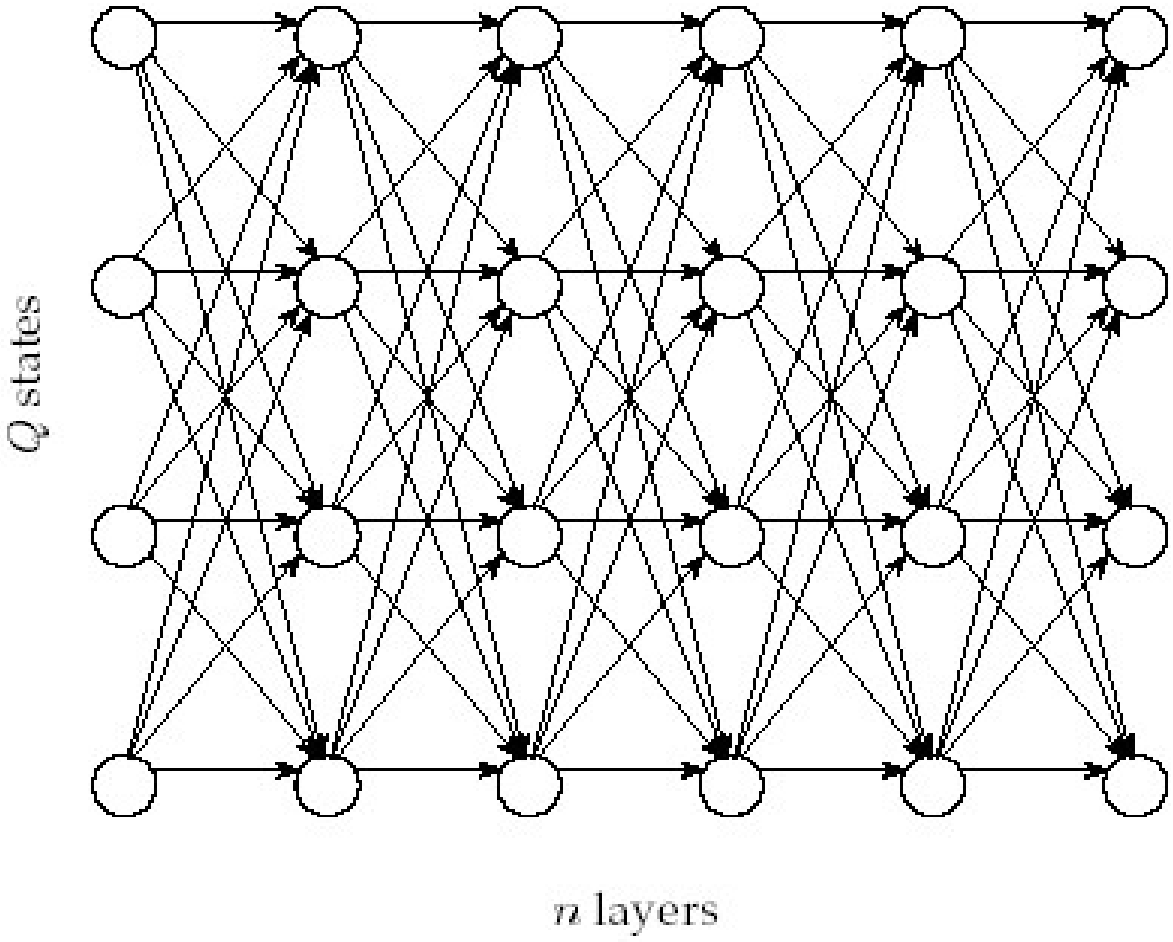
Decoding Problem

- **Goal:** Find an optimal hidden path of states given observations.
- **Input:** Sequence of observations $x = x_1 \ldots x_n$ generated by an HMM $M(\Sigma, Q, A, E)$
- **Output:** A path that maximizes $P(x|\pi)$ over all possible paths π.

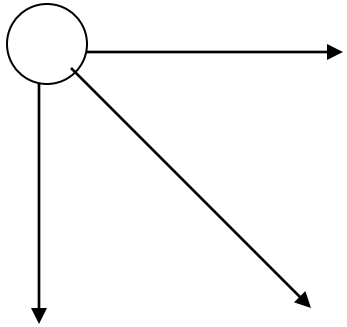
Building Manhattan for Decoding Problem

- Andrew Viterbi used the Manhattan grid model to solve the *Decoding Problem*.
- Every choice of $\pi = \pi_1 \ldots \pi_n$ corresponds to a path in the graph.
- The only valid direction in the graph is *eastward.*
- This graph has $|Q|^2(n-1)$ edges.

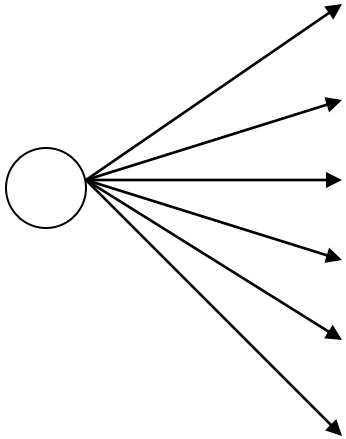
Edit Graph for Decoding Problem



Decoding Problem vs. Alignment Problem



Valid directions in the *alignment problem.*



Valid directions in the *decoding problem.*

Decoding Problem as Finding a Longest Path in a DAG

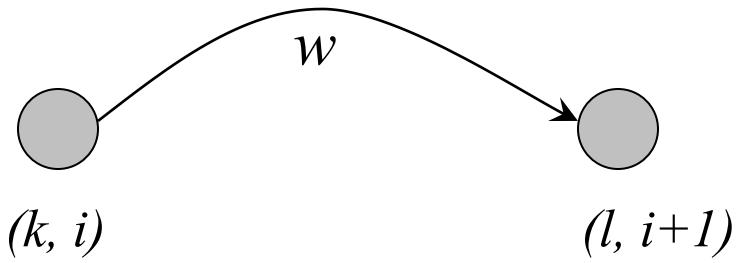
- The *Decoding Problem* is reduced to finding a longest path in the *directed acyclic graph (DAG)* above.

- **Notes:** the length of the path is defined as the *product* of its edges' weights, not the *sum.*

Decoding Problem (cont'd)

- Every path in the graph has the probability $P(x|\pi)$.
- The Viterbi algorithm finds the path that maximizes $P(x|\pi)$ among all possible paths.
- The Viterbi algorithm runs in $\boldsymbol{O(n|Q|^2)}$ time.

Decoding Problem: weights of edges

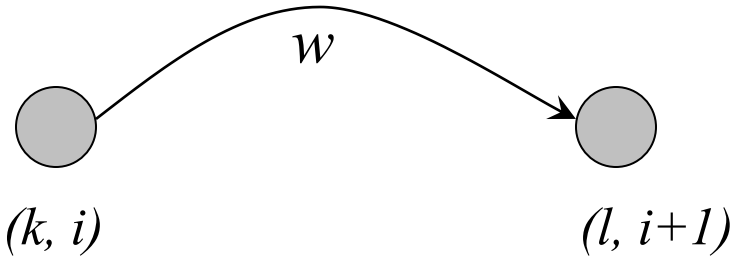


The weight ***w*** is given by:

???

Decoding Problem: weights of edges

$$P(x|\pi) = \prod_{i=0}^{n} e_{\pi_{i+1}}(x_{i+1}) \cdot a_{\pi_i, \pi_{i+1}}$$

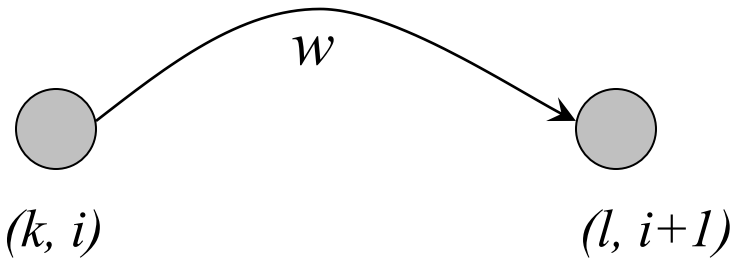


The weight ***w*** is given by:

??

Decoding Problem: weights of edges

i-th term = $e_{\pi_{i+1}}(x_{i+1}) \cdot a_{\pi_i, \pi_{i+1}}$

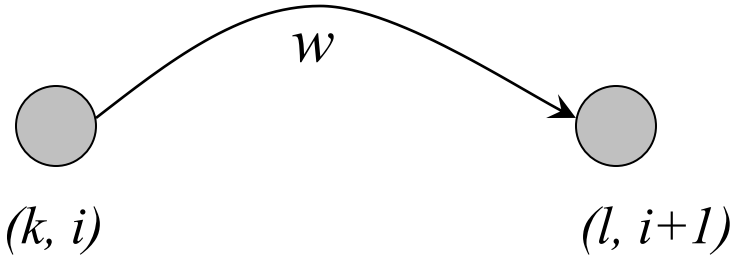


The weight ***w*** is given by:

?

Decoding Problem: weights of edges

i-th term = $e_{\pi_i}(x_i) \cdot a_{\pi_i, \pi_{i+1}} = \mathbf{e_l(x_{i+1}) \cdot a_{kl}}$ *for* $\pi_i = k, \pi_{i+1} = l$



The weight $\mathbf{w = e_l(x_{i+1}) \cdot a_{kl}}$

Decoding Problem and Dynamic Programming

$s_{l,i+1} = \max_{k \in Q} \{s_{k,i} \cdot \text{weight of edge between } (k,i) \text{ and } (l,i+1)\} =$

$\max_{k \in Q} \{s_{k,i} \cdot a_{kl} \cdot e_l(x_{i+1})\} =$

$e_l(x_{i+1}) \cdot \max_{k \in Q} \{s_{k,i} \cdot a_{kl}\}$

Decoding Problem (cont'd)

- Initialization:
  - $s_{begin,0} = 1$
  - $s_{k,0}$ = 0 for $k \neq begin$.
- Let π^* be the optimal path. Then,

$$P(x|\pi^*) = \max_{k \in Q} \{s_{k,n} \cdot a_{k,end}\}$$

Decoding Problem (cont'd)

- Initialization:
  - $s_{begin,0} = 1$
  - $s_{k,0}$ = 0 for $k \neq begin$.
- Let π^* be the optimal path. Then,

$$P(x|\pi^*) = \max_{k \in Q} \{s_{k,n} \cdot a_{k,end}\}$$

Is there a problem here?

Viterbi Algorithm

- The value of the product can become extremely small, which leads to overflowing.

Viterbi Algorithm

- The value of the product can become extremely small, which leads to overflowing.
- To avoid overflowing, use log value instead.

$$s_{k,i+1} = \log e_l(x_{i+1}) + \max_{k \in Q} \{s_{k,i} + \log(a_{kl})\}$$

FORWARD/BACKWARD

Forward-Backward Problem

Given: a sequence of coin tosses generated by an HMM.

Goal: find the probability that the dealer was using a biased coin at a particular time.

Forward Algorithm

- Define $f_{k,i}$ (*forward probability*) as the probability of emitting the prefix $x_1 \ldots x_i$ and reaching the state $\pi = k$.
- The recurrence for the forward algorithm:

$$f_{k,i} = e_k(x_i) \cdot \sum_{l \in Q} f_{l,i-1} \cdot a_{lk}$$

Backward Algorithm

- However, *forward probability* is not the only factor affecting $P(\pi_i = k|x)$.

- The sequence of transitions and emissions that the HMM undergoes between π_{i+1} and π_n also affect $P(\pi_i = k|x)$.

Backward Algorithm (cont'd)

- Define *backward probability* $b_{k,i}$ as the probability of being in state $\pi_i = k$ and emitting the *suffix* $x_{i+1} \ldots x_n$.
- The recurrence for the *backward algorithm*:

$$b_{k,i} = \sum_{l \in Q} e_l(x_{i+1}) \cdot b_{l,i+1} \cdot a_{kl}$$

Forward-Backward Algorithm

- The probability that the dealer used a biased coin at any moment i:

$$P(\pi_i = k|x) = \frac{P(x, \pi_i = k)}{P(x)} = \frac{f_k(i) \,.\, b_k(i)}{P(x)}$$

P(x) is the sum of $P(x, \pi_i = k)$ over all *k*

PROFILE HMM

Finding Distant Members of a Protein Family

- A distant cousin of functionally related sequences in a protein family may have weak pairwise similarities with each member of the family and thus fail significance test.
- However, they may have weak similarities with ***many*** members of the family.
- The goal is to align a sequence to ***all*** members of the family at once.
- Family of related proteins can be represented by their multiple alignment and the corresponding profile.

Profile Representation of Protein Families

Aligned DNA sequences can be represented by a $4 \cdot n$ profile matrix reflecting the frequencies of nucleotides in every aligned position.

A	.72	.14	0	0	.72	.72	0	0
T	.14	.72	0	0	0	.14	.14	.86
G	.14	.14	.86	.44	0	.14	0	0
C	0	0	.14	.56	.28	0	.86	.14

Protein family can be represented by a $20 \cdot n$ profile representing frequencies of amino acids.

Profiles and HMMs

- HMMs can also be used for aligning a sequence against a profile representing protein family.
- A $20 \cdot n$ profile P corresponds to n sequentially linked *match* states $M_1, \ldots, M_n$ in the profile HMM of P.

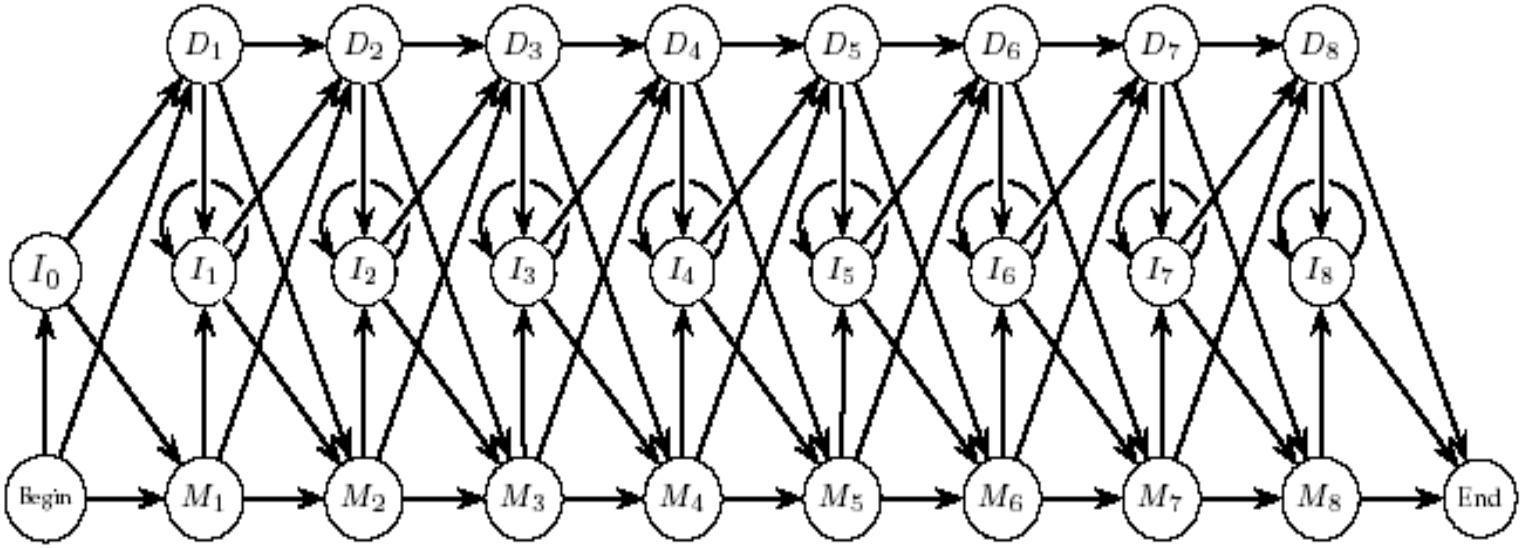
Multiple Alignments and Protein Family Classification

- Multiple alignment of a protein family shows variations in conservation along the length of a protein
- Example: after aligning many globin proteins, the biologists recognized that the helices region in globins are more conserved than others.

What are Profile HMMs ?

- A Profile HMM is a probabilistic representation of a multiple alignment.
- A given multiple alignment (of a protein family) is used to build a profile HMM.
- This model then may be used to find and score less obvious potential matches of new protein sequences.

Profile HMM



A profile HMM

Building a profile HMM

- Multiple alignment is used to construct the HMM model.
- Assign each column to a *Match* state in HMM. Add I*nsertion* and *Deletion* state.
- Estimate the emission probabilities according to amino acid counts in column. Different positions in the protein will have different emission probabilities.
- Estimate the transition probabilities between *Match, Deletion* and *Insertion* states
- The HMM model gets trained to derive the optimal parameters.

States of Profile HMM

- Match states $M_1 \ldots M_n$ (plus *begin/end* states)
- Insertion states $I_0 I_1 \ldots I_n$
- Deletion states $D_1 \ldots D_n$

Transition Probabilities in Profile HMM

- $log(a_{MI})+log(a_{IM})$ = gap initiation penalty
- $log(a_{II})$ = gap extension penalty

Emission Probabilities in Profile HMM

- Probabilty of emitting a symbol *a* at an insertion state I_j:

$$e_{Ij}(a) = p(a)$$

where *p(a)* is the frequency of the occurrence of the symbol *a* in all the sequences.

Profile HMM Alignment

- Define $v^{M}_{j}(i)$ as the logarithmic likelihood score of the best path for matching $x_1..x_i$ to profile HMM ending with x_i emitted by the state M_j.

- $v^{I}_{j}(i)$ and $v^{D}_{j}(i)$ are defined similarly.

Profile HMM Alignment: Dynamic Programming

$$v^M_j(i) = \log\left(e_{M_j}(x_i)/p(x_i)\right) + \max \begin{cases} v^M_{j-1}(i-1) + \log(a_{M_{j-1},M_j}) \\ v^I_{j-1}(i-1) + \log(a_{I_{j-1},M_j}) \\ v^D_{j-1}(i-1) + \log(a_{D_{j-1},M_j}) \end{cases}$$

$$v^I_j(i) = \log\left(e_{I_j}(x_i)/p(x_i)\right) + \max \begin{cases} v^M_j(i-1) + \log(a_{M_j,\ I_j}) \\ v^I_j(i-1) + \log(a_{I_j,\ I_j}) \\ v^D_j(i-1) + \log(a_{D_j,\ I_j}) \end{cases}$$