HIDDEN MARKOV MODELS

Outline

- CG-islands
- The "Fair Bet Casino"
- Hidden Markov Model
- Decoding Algorithm
- Forward-Backward Algorithm
- Profile HMMs
- HMM Parameter Estimation
- Viterbi training
- Baum-Welch algorithm

CG-Islands (=*CpG islands*)

- Given 4 nucleotides: probability of occurrence is ~ 1/4. Thus, probability of occurrence of a dinucleotide is ~ 1/16.
- However, the frequencies of dinucleotides in DNA sequences vary widely.
- In particular, *CG* is typically underrepresented (frequency of *CG* is typically < 1/16)

Why *CG*-Islands?

- *CG* is the least frequent dinucleotide because *C* in *CG* is easily *methylated and* has the tendency to mutate into T afterwards
- However, the methylation is suppressed around genes in a genome. So, *CG* appears at relatively high frequency within these *CG* islands
- So, finding the *CG* islands in a genome is an important problem

CG Islands and the "Fair Bet Casino"

- The *CG* islands problem can be modeled after a problem named ***"The Fair Bet Casino"***
- The game is to flip coins, which results in only two possible outcomes: **H**ead or **T**ail.
- The **F**air coin will give **H**eads and **T**ails with same probability ½.
- The **B**iased coin will give **H**eads with prob. ¾.

The “Fair Bet Casino” (cont’d)

- Thus, we define the probabilities:
  - $P(H|F) = P(T|F) = ½$
  - $P(H|B) = ¾$, $P(T|B) = ¼$
  - The crooked dealer changes between Fair and Biased coins with probability 10%

The Fair Bet Casino Problem

- **Input:** A sequence $x = x_1x_2x_3\ldots x_n$ of coin tosses made by two possible coins (***F*** or ***B***).
- **Output:** A sequence $\pi = \pi_1\ \pi_2\ \pi_3\ldots\ \pi_n$, with each π_i being either *F* or *B* indicating that x_i is the result of tossing the Fair or Biased coin respectively.

Problem…

Fair Bet Casino Problem

Any observed outcome of coin tosses could have been generated by any sequence of states!

Need to incorporate a way to grade different sequences differently.

↓

Decoding Problem

P(x| fair coin) vs. P(x| biased coin)

- Suppose first that dealer never changes coins. Some definitions:
  - P(x|fair coin): prob. of the dealer using the *F* coin and generating the outcome x.
  - P(x|biased coin): prob. of the dealer using the *B* coin and generating outcome x.

P(x| fair coin) vs. P(x| biased coin)

- P(x|fair coin)=P($x_1 \ldots x_n$|fair coin)

$$\Pi_{i=1,n}\, p\,(x_i|\text{fair coin}) = (1/2)^n$$

- P(x|biased coin)= P($x_1 \ldots x_n$|biased coin)=

$$\Pi_{i=1,n}\, p\,(x_i|\text{biased coin}) = (3/4)^k (1/4)^{n-k} = 3^k/4^n$$

  - k - the number of ***H***eads in x.

P(x| fair coin) vs. P(x| biased coin)

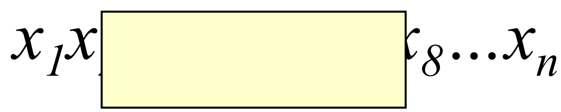
- P(x|fair coin) = P(x|biased coin)
- $1/2^n = 3^k/4^n$
- $2^n = 3^k$
- $n = k \log_2 3$
- when $k = n / \log_2 3$ ($k \sim 0.67n$)

Log-odds Ratio

- We define ***log-odds ratio*** as follows:

$$\log_2(P(x|\text{fair coin}) / P(x|\text{biased coin}))$$
$$= \Sigma^k_{i=1} \log_2(p^+(x_i) / p^-(x_i))$$
$$= n - k \log_2 3$$

Computing Log-odds Ratio in Sliding Windows



Consider a *sliding window* of the outcome sequence. Find the log-odds for this short window.

Disadvantages:
- the length of CG-island is not known in advance
- different windows may classify the same position differently

Hidden Markov Model (HMM)

- Can be viewed as an abstract machine with *k hidden* states that emits symbols from an alphabet Σ.
- Each state has its own probability distribution, and the machine switches between states according to this probability distribution.
- While in a certain state, the machine makes 2 decisions:
  - What state should I move to next?
  - What symbol - from the alphabet Σ - should I emit?

Why "Hidden"?

- Observers can see the emitted symbols of an HMM but have *no ability to know which state the HMM is currently in*.
- Thus, the goal is to infer the most likely hidden states of an HMM based on the given sequence of emitted symbols.

HMM Parameters

Σ: set of emission characters.

Ex.: $\Sigma = \{H, T\}$ for coin tossing

$\Sigma = \{1, 2, 3, 4, 5, 6\}$ for dice tossing

Q: set of hidden states, each emitting symbols from Σ.

$Q=\{F,B\}$ for coin tossing

HMM Parameters (cont'd)

$A = (a_{kl})$: a |Q| x |Q| matrix of probability of changing from state *k* to state *l*.

$$a_{FF} = 0.9 \qquad a_{FB} = 0.1$$

$$a_{BF} = 0.1 \qquad a_{BB} = 0.9$$

$E = (e_k(b))$: a |Q| x |Σ| matrix of probability of emitting symbol *b* while being in state *k*.

$$e_F(0) = \tfrac{1}{2} \qquad e_F(1) = \tfrac{1}{2}$$

$$e_B(0) = \tfrac{1}{4} \qquad e_B(1) = \tfrac{3}{4}$$

HMM for Fair Bet Casino

- The *Fair Bet Casino* in *HMM* terms:

  $\Sigma = \{0, 1\}$ (0 for ***T***ails and 1 ***H***eads)

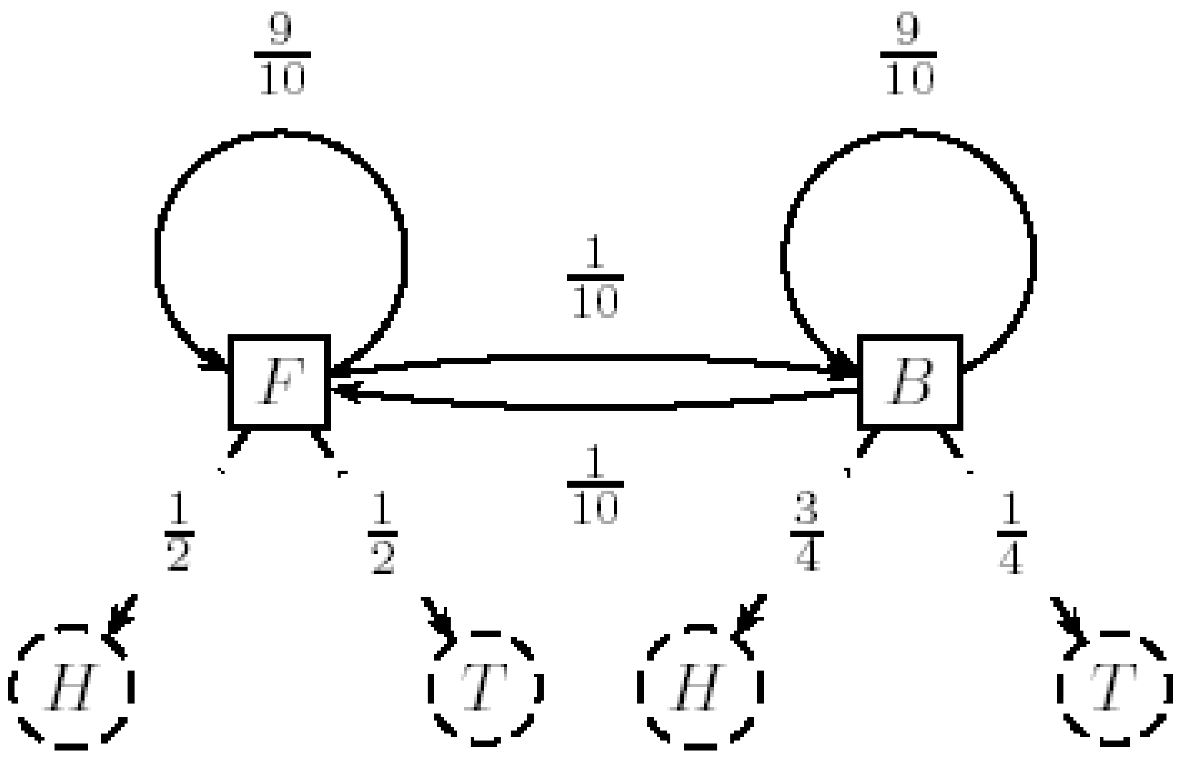
  Q = {*F*,*B*} – *F* for Fair & *B* for Biased coin.

- Transition Probabilities *A* *** Emission Probabilities *E*

	Fair	Biased
Fair	$a_{FF} = 0.9$	$a_{FB} = 0.1$
Biased	$a_{BF} = 0.1$	$a_{BB} = 0.9$

	Tails(0)	Heads(1)
Fair	$e_F(0) = ½$	$e_F(1) = ½$
Biased	$e_B(0) = ¼$	$e_B(1) = ¾$

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

Probability that x_i was emitted from state π_i

x	0	1	0	1	1	1	0	1	0	0	1
π =	F	F	F	B	B	B	B	B	F	F	F
$P(x_i \mid \pi_i)$	½	½	½	¾	¾	¾	¼	¾	½	½	½
$P(\pi_{i-1} \rightarrow \pi_i)$	½	$^9/_{10}$	$^9/_{10}$	$^1/_{10}$	$^9/_{10}$	$^9/_{10}$	$^9/_{10}$	$^9/_{10}$	$^1/_{10}$	$^9/_{10}$	$^9/_{10}$

Transition probability from state π_{i-1} to state π_i

$P(x|\pi)$ Calculation

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

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

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

$P(x|\pi)$ Calculation

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

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

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

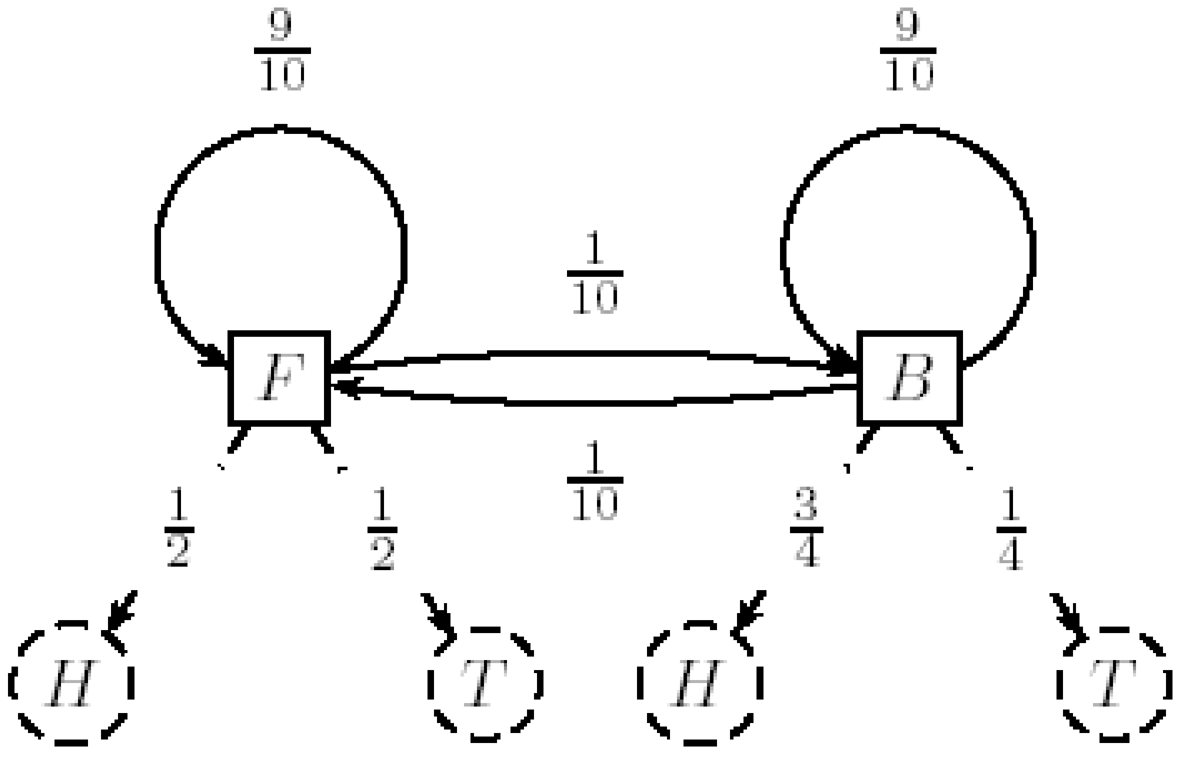
$$= \prod e_{\pi_{i+1}}(x_{i+1}) \cdot a_{\pi_i, \pi_{i+1}}$$

if we count from $i=0$ instead of $i=1$

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.

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

Probability that x_i was emitted from state π_i

x	0	1	0	1	1	1	0	1	0	0	1
π =	F	F	F	B	B	B	B	B	F	F	F
$P(x_i \mid \pi_i)$	½	½	½	¾	¾	¾	¼	¾	½	½	½
$P(\pi_{i-1} \rightarrow \pi_i)$	½	$^9/_{10}$	$^9/_{10}$	$^1/_{10}$	$^9/_{10}$	$^9/_{10}$	$^9/_{10}$	$^9/_{10}$	$^1/_{10}$	$^9/_{10}$	$^9/_{10}$

Transition probability from state π_{i-1} to state π_i

$P(x|\pi)$ Calculation

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

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

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

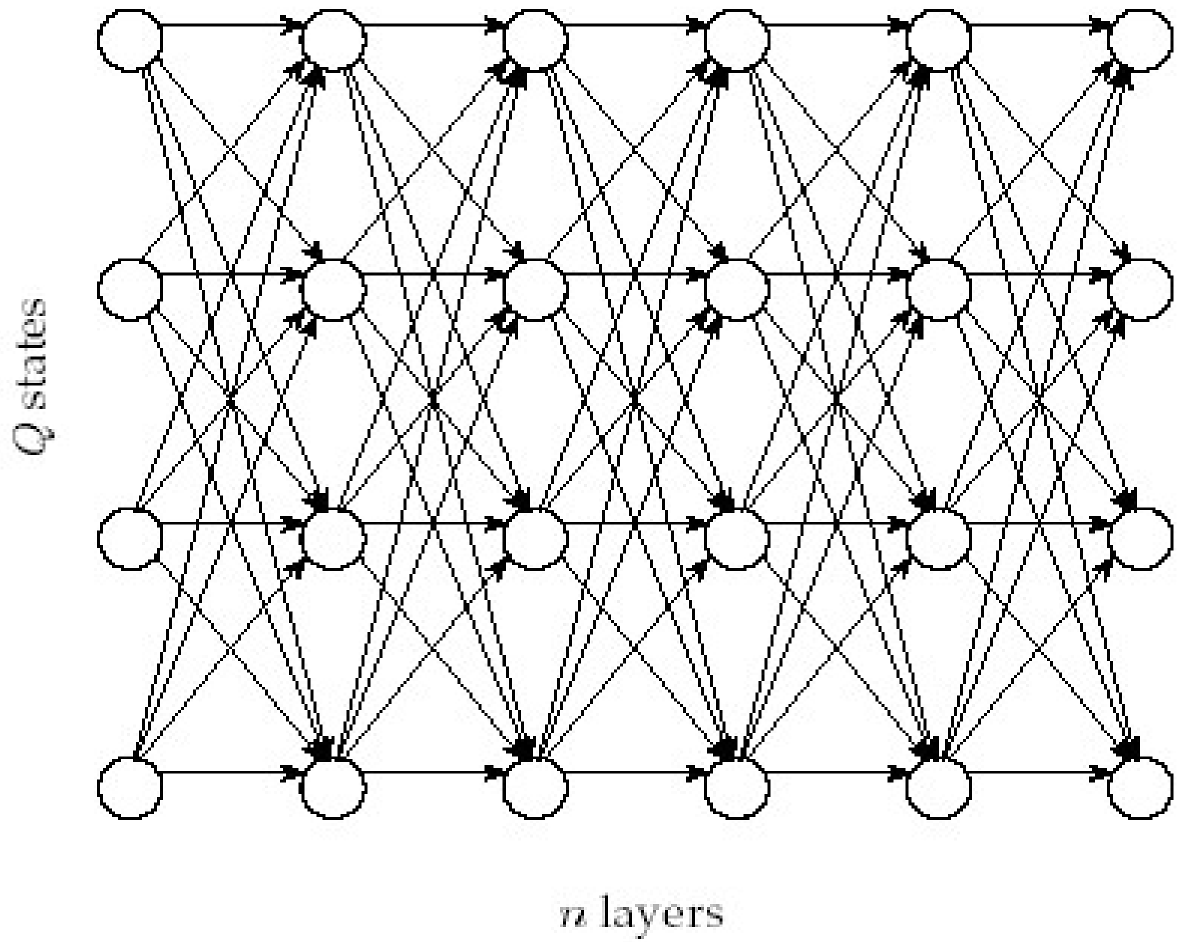
Decoding Problem

- **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.

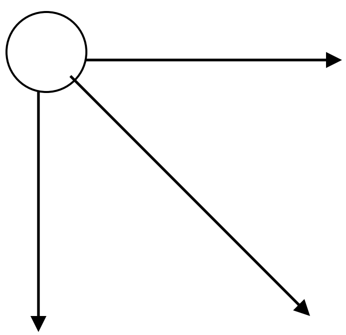
Building Manhattan for Decoding Problem

- 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.

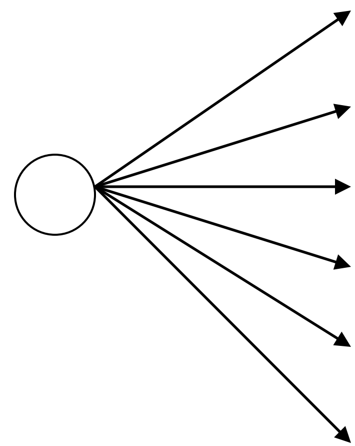
Edit Graph for Decoding Problem



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

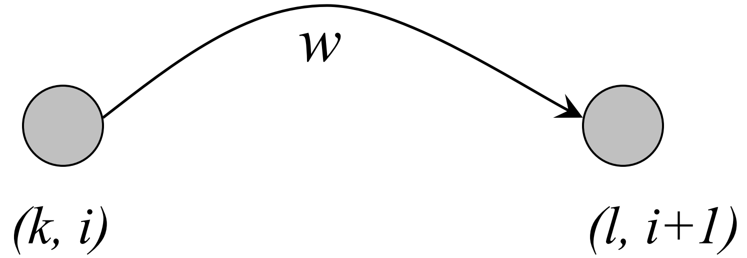
- The *Decoding Problem* is reduced to finding a longest path in the *directed acyclic graph (DAG)* above.

- **<u>Notes:</u>** 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

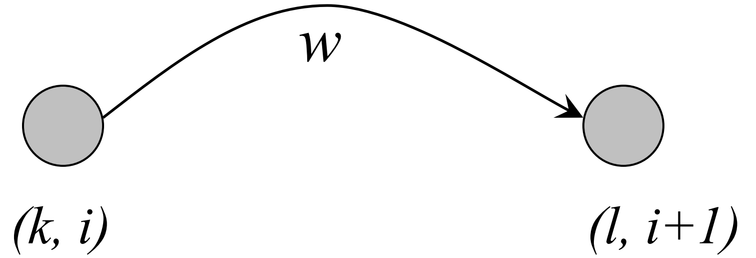


The weight ***w*** is given by:

???

Decoding Problem: weights of edges

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

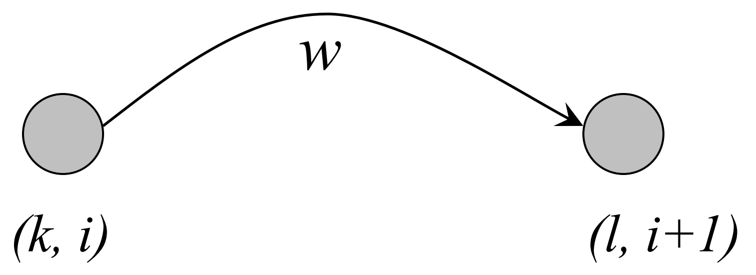


The weight ***w*** is given by:

??

Decoding Problem: weights of edges

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

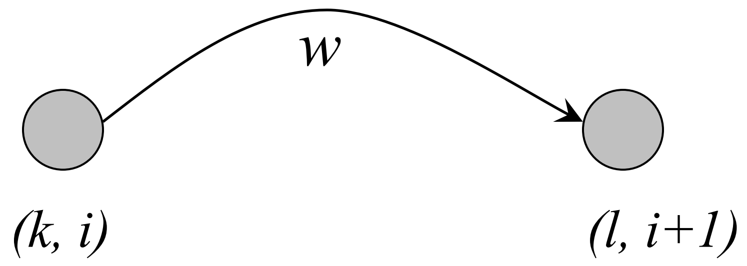


The weight ***w*** is given by:

?

Decoding Problem: weights of edges

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



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

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.

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