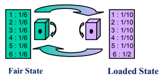
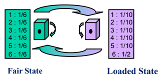
Hidden Markov Models

Seven

Introduction to Bioinformatics

Sami Khuri
Department of Computer Science
San José State University
San José, CA 95192
June 2016





Hidden Markov Models

Russian Mathematician
Saint Petersburg
1856 - 1922

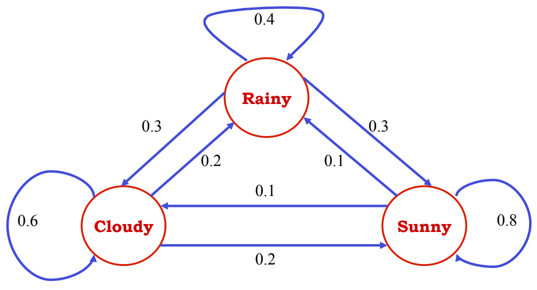
- Andrei Andreyevich Markov
- Markov Chain
- Homology Model
- profile Hidden Markov Model
- Viterbi Algorithm
- Forward Algorithm
- Backward Algorithm
- EM Algorithm

Three-State Markov Weather Model

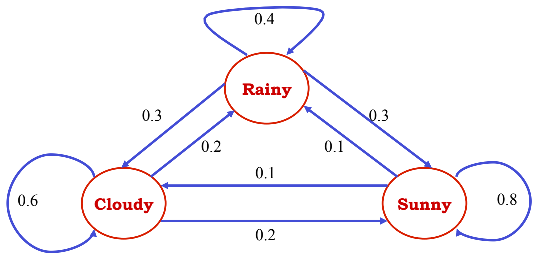
- We have three states:
  Rainy (R) Cloudy (C) Sunny (S)
- The weather on any day t is characterized by a single state.
- State transition probability matrix:

$$\mathbf{A} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

Markov Weather Model



Markov Weather Model



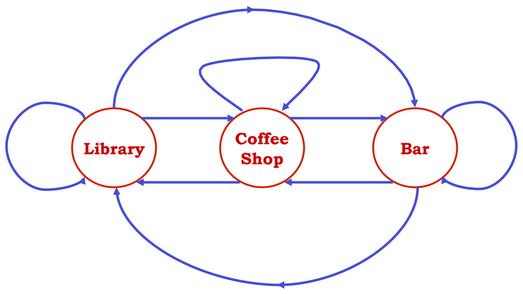
Compute the probability of observing *SSRRSCS* given that today it is sunny (i.e., we are in state *S)*.

Solving the Weather Example

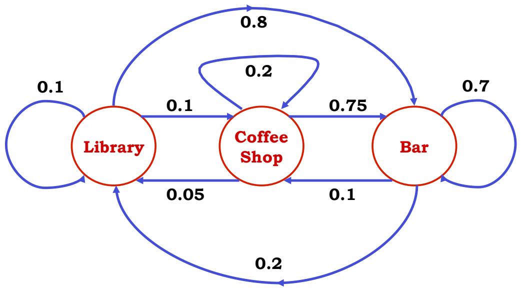
- Observation sequence: $O = (S, S, S, R, R, S, C, S)$
- Using the chain rule we get:

$$\begin{aligned} &P(O \mid \text{model}) \\ &= P(S,S,S,R,R,S,C,S \mid \text{model}) \\ &= P(S)P(S \mid S)P(S \mid S)P(R \mid S)P(R \mid R)\times \\ &\quad P(S \mid R)P(C \mid S)P(S \mid C) \\ &= \pi a_{33}a_{33}a_{31}a_{11}a_{13}a_{32}a_{23} \\ &= (1)(0.8)^2(0.1)(0.4)(0.3)(0.1)(0.2) = 1.536\times 10^{-4} \end{aligned}$$

States and Transitions



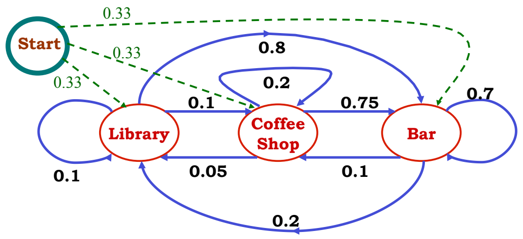
The Distressed Student Model



Evaluating Observations

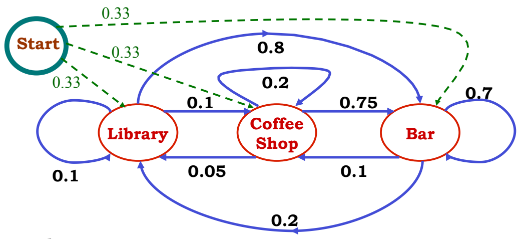
- The probability of observing a given sequence is equal to the product of all observed **transition probabilities**.
- Suppose that:
  - **L**: student is in state Library
  - **C**: student is in state Coffee Shop
  - **B**: student is in state Bar

Starting State of the Student



The Model has a Start State with transition probabilities of going to L, C, or B of 1/3.

Behavior of Three Students



Student 1 : LLLCBCLLBBLL
Student 2 : LCBLBBCBBBBL
Student 3 : CCCLCCCBCCCL

Computing Observed Sequences

- The probability of observing a given sequence is equal to the product of all observed transition probabilities.
- Example:
- P(LLLCBCLLBBLL)

  = 1/3 * 0.1 * 0.1 * 0.1 * 0.75 * 0.1 * 0.05 * 0.1 * 0.8 * 0.7 * 0.2 * 0.1

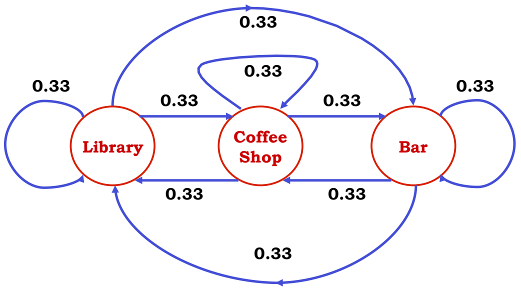
  = $1.4 * 10^{-9}$

Computing Observed Sequences

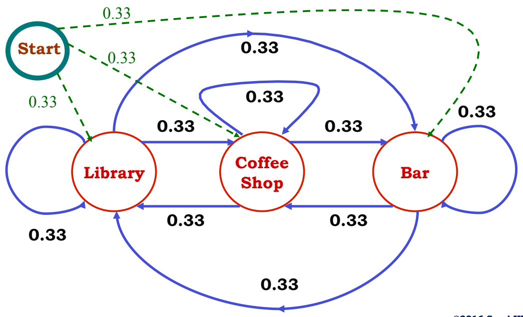
- P(LCBLBBCBBBBL)
  = 1/3 * 0.1 * 0.75 * 0.2 * 0.8 * 0.7 * 0.1 * 0.75 * 0.7 * 0.7 * 0.7 * 0.2
  = $1.4406 * 10^{-5}$

- P(CCCLCCCBCCCL)
  = 1/3 * 0.2 * 0.2 * 0.05 * 0.1 * 0.2 * 0.2 * 0.75 * 0.1 * 0.2 * 0.2 * 0.05
  = $4 * 10^{-10}$

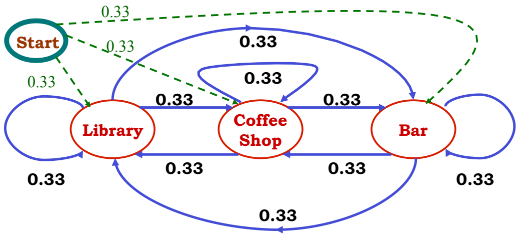
The Random Model The Null Hypothesis



Start State with Random Model



Students with Random Model



Student 1 : LLLCBCLLBBLL = 1.8817×10^{-6}
Student 2 : LCBLBBCBBBBL = 1.8817×10^{-6}
Student 3 : CCCLCCCBCCCL = 1.8817×10^{-6}

Odds and Log Ratios

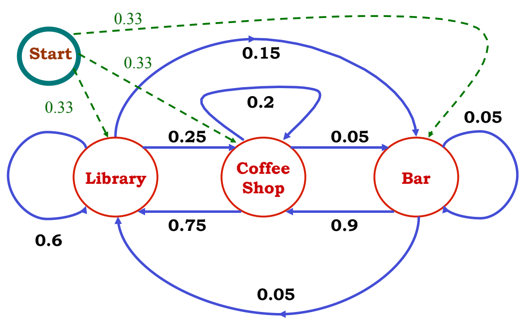
- To determine the significance of the results obtained with the 3 students, compare them to the null model (random model)
- Odds Ratio =
  P(x | Distressed Model) / P(x | Null Model)
- Log Odds =
  Log [P(x | Distressed Model) / P(x | Null Model)]

Likelihood Ratios: Distressed

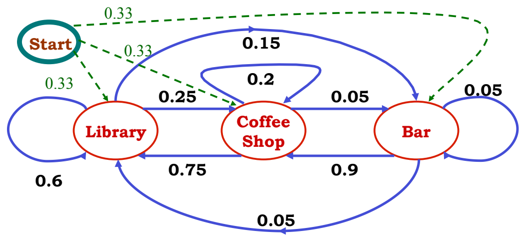
- **Likelihood ratios**:
  Student 1 : = $1.4 \times 10^{-9} / 1.8817 \times 10^{-6}$ = x
  Student 2 : = $1.4406 \times 10^{-5} / 1.8817 \times 10^{-6}$ = y
  Student 3 : = $4 \times 10^{-10} / 1.8817 \times 10^{-6}$ = z

- **Log likelihood ratios**:
  Student 1 = log x = - 10.39
  Student 2 = log y = 2.94
  Student 3 = log z = - 12.20

The Successful Student Model



Students with Successful Model



Student 1 : LLLCBCLLBBLL
Student 2 : LCBLBBCBBBBL
Student 3 : CCCLCCCBCCCL

Outcomes with Successful Model

- P(LLLCBCLLBBLL)
  = 1/3 * 0.6 * 0.6 * 0.25 * 0.05 * 0.9 * 0.75 * 0.6 * 0.15 * 0.05 * 0.05 * 0.6 = 1.3669 * 10^{-7}
- P(LCBLBBCBBBBL)
  = 1/3 * 0.25 * 0.05 * 0.05 * 0.15 * 0.05 * 0.9 * 0.05 * 0.05 * 0.05 * 0.05 * 0.05 = 4.3945 * 10^{-13}
- P(CCCLCCCBCCCL)
  = 1/3 * 0.2 * 0.2 * 0.75 * 0.25 * 0.2 * 0.2 * 0.05 * 0.9 * 0.2 * 0.2 * 0.75 = 1.35 * 10^{-7}

Likelihood Ratios: Successful

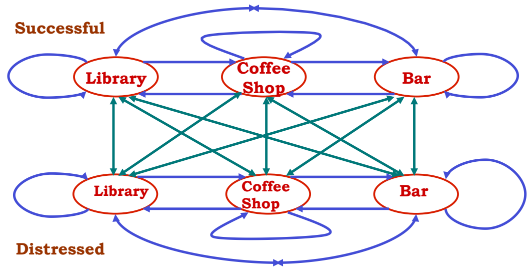
- **Likelihood ratios**:

  Student 1 : $= 1.3669 \times 10^{-7} / 1.8817 \times 10^{-6} = x$
  Student 2 : $= 4.3945 \times 10^{-13} / 1.8817 \times 10^{-6} = y$
  Student 3 : $= 1.35 \times 10^{-7} / 1.8817 \times 10^{-6} = z$

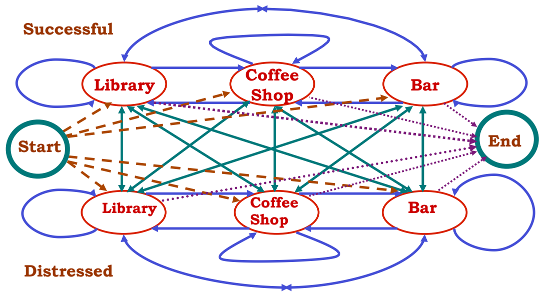
- **Log likelihood ratios**:

Student 1 =	log x	= - 3.78
Student 2 =	log y	= - 22.03
Student 3 =	log z	= - 3.8

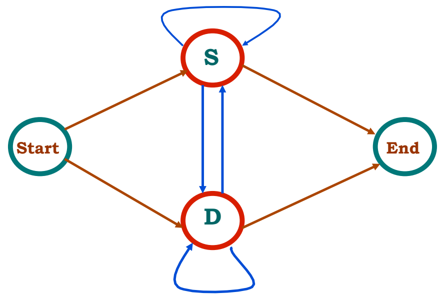
HMM – Combined Model



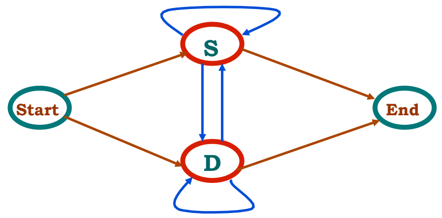
HMM – Combined Model



Hidden Markov Model



Evaluating Hidden States



Given an observation: LLLCBCLBBCL, find the sequence of states which is the most likely to have produced the observation.

Models of Sequences

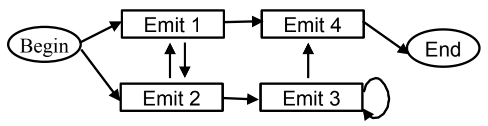
- Consists of states (circles) and transitions (arcs) labelled with probabilities.
- States have probabilities of "emitting" an element of a sequence (or nothing).
- Arcs have transitional probabilities of moving from one state to another.
  - Sum of probabilities of arcs out of a state must be 1
  - Self-loops are allowed.

Markov Chain

- A sequence is said to be **Markovian** if the probability of the occurrence of an element in a particular position depends only on the **previous** elements in the sequence.
- Order of a Markov chain depends on how many previous elements influence the probability:
  - **0th order**: uniform probability at every position
  - **1st order**: probability depends only on the immediately previous position.

Simple Markov Model

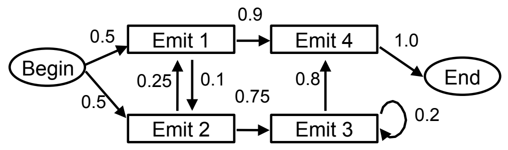
- **Example**: Each state emits (or, equivalently, recognizes) a particular number with probability 1, and each transition is equally likely.



Possible sequences: 1234 234 14
121214 2123334

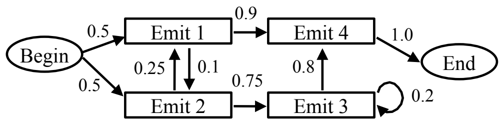
Probabilistic Markov Model

Now, add probabilities to each transition



Probabilistic Markov Model

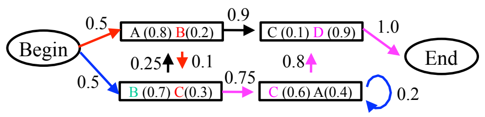
We can compute the probability of occurrence of any output sequence:



p (**1234**) = 0.5 * 0.1 * 0.75 * 0.8 = **0.03**
p (**14**) = 0.5 * 0.9 = **0.45**
p (**2334**) = 0.5 * 0.75 * 0.2 * 0.8 = **0.06**

Probabilistic Emission

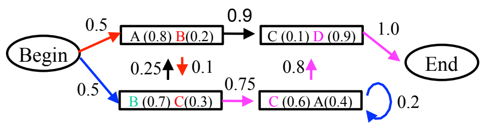
- Define a set of **emission probabilities** for elements in the states.
- Given an output sequence, where does it come from?



BCCD or BCCD ?

Hidden Markov Models

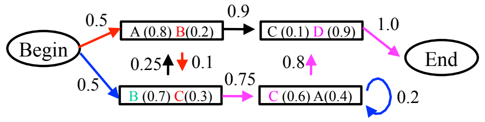
- Emission uncertainty means the sequence does not identify a unique path.
- The states are "**hidden**":



BCCD or **BCCD** ?

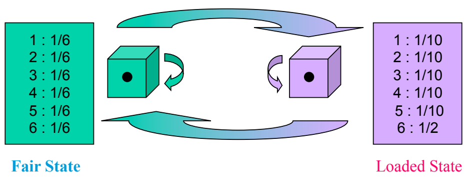
Computing Probabilities

Probability of an output sequence is the sum of all the paths that can produce it:



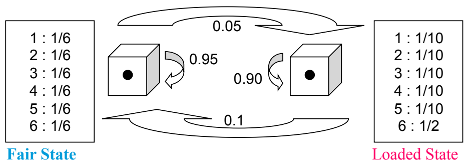
p(BCCD) = (0.5 * 0.2 * 0.1 * 0.3 * 0.75 * 0.6 * 0.8 * 0.9)
+ (0.5 * 0.7 * 0.75 * 0.6 * 0.2 * 0.6 * 0.8 * 0.9)
= 0.000972 + 0.013608 = 0.01458

The Dishonest Casino (I)



If we see a sequence of rolls (the sequence of observations) we do not know which rolls used a loaded die and which used a fair die.

The Dishonest Casino (II)



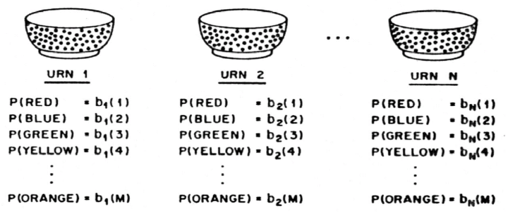
$$\Pi = \begin{pmatrix} 0.8 \\ 0.2 \end{pmatrix} \qquad A = \begin{pmatrix} 0.95 & 0.05 \\ 0.1 & 0.9 \end{pmatrix} \qquad B = \begin{pmatrix} 0.16 & 0.16 & 0.16 & 0.16 & 0.16 & 0.16 \\ 0.10 & 0.10 & 0.10 & 0.10 & 0.10 & 0.50 \end{pmatrix}$$

Initial State **State Transitions** **Emissions**

The Urn and Ball Model (I)

- N urns containing colored balls
- M distinct colors of balls
- Algorithm that generates Observed Sequence:
  1. Pick initial urn according to some random process.
  2. Randomly pick a ball from the chosen urn, record its color and then put it back.
  3. Randomly pick an urn
  4. Repeat steps 2 and 3

The Urn and Ball Model (II)



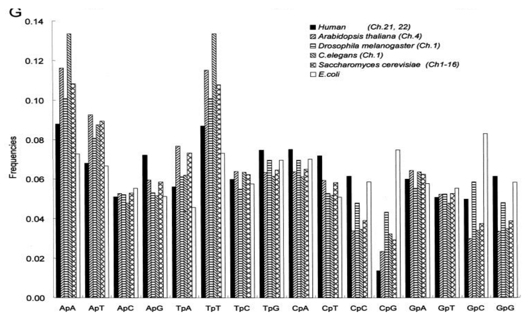
An urn is selected and then a ball is selected from the urn, its color is recorded, and the ball is put back in the urn.
Given the sequence of **observed colors**, can we guess from which urn each ball comes from?

Looking for CpG Islands

Example:

- A CpG island in humans refers to the dinucleotide CG and not the basepair CG.
- The C of CpG is generally methylated to inactivate genes hence CpG is found around "start" regions of many genes more often than elsewhere.
- Methylated C is easily mutated into T.

The Rarity CpG Islands



CpG Island Criteria

- According to Gardiner-Garden and Fromer, CpG islands are commonly defined as regions of DNA
  - of at least 200 bp in length,
  - that have a G+C content above 50%
  - that have a ratio of observed vs. expected CpGs close to or above 0.6.
- Sets of CG repeat elements, usually found upstream of transcribed regions of the genome.

Looking for CpG Islands

- CpG islands are therefore rare in other locations
- CpG islands are generally a few hundred base pairs long

Questions:

1. Given a short DNA fragment, does it come from a **CpG island** or not?
2. Given a long unannotated sequence of DNA, how do we find the **CpG islands**?

Building an HMM for CpG Islands

- A set of human sequences were considered and 48 CpG islands were tabulated.
- Two Markov chain models were built:
  - One for the regions labeled as CpG islands (the '+' model or Model 1)
  - One for the remainder of the sequences (the '-' model or Model 2).

Transition Probabilities

The transition probabilities of each model were computed by:

$$a_{st}^{+} = \frac{c_{st}^{+}}{\sum_{t'} c_{st'}^{+}} \qquad a_{st}^{-} = \frac{c_{st}^{-}}{\sum_{t'} c_{st'}^{-}}$$

c_{st}^{+} is the number of times letter t followed letter s in the plus model.

The Two Transition Tables

Markov chain Model 1
Transition Frequencies within 48 putative CpG islands in humans

+	A	C	G	T
A	0.180	0.274	0.426	0.120
C	0.171	0.368	0.274	0.188
G	0.161	0.339	0.375	0.125
T	0.079	0.355	0.384	0.182

Markov chain Model 2
Transition frequencies in Non CpG island DNA

-	A	C	G	T
A	0.300	0.205	0.285	0.210
C	0.322	0.298	0.078	0.302
G	0.248	0.246	0.298	0.208
T	0.177	0.239	0.292	0.292

Log Odds Ratio

- Given any sequence, we compute the **log-odds ratio** to discriminate between the two models :

$$S(x) = \log \frac{P(x \mid the + \text{model})}{P(x \mid the - \text{model})} = \sum_{i=1}^{L} \log \frac{a_{x_{i-1}x_i}^{+}}{a_{x_{i-1}x_i}^{-}}$$

- S(x)>0 means x is likely to be a CpG island.
- The ratio is also called the **log likelihood ratio** of transition probabilities.

Log Likelihood Ratios

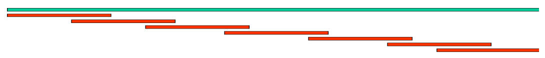
$$S(x) = \log \frac{P(x \mid the + \text{model})}{P(x \mid the - \text{model})} = \sum_{i=1}^{L} \log \frac{a_{x_{i-1}x_i}^{+}}{a_{x_{i-1}x_i}^{-}}$$

Log models	A	C	G	T
A	-0.740	0.417	0.580	-0.803
C	-0.913	0.302	1.812	-0.685
G	-0.624	0.461	0.331	-0.730
T	-1.169	0.573	0.393	-0.679

The table's unit is the bit since base 2 is used for the computation of the individual entries of the table.

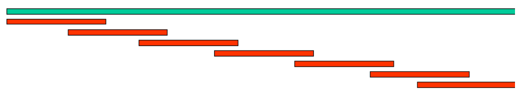
Looking for CpG Islands

- Given a long unannotaded sequence of DNA, how do we find the **CpG islands**?



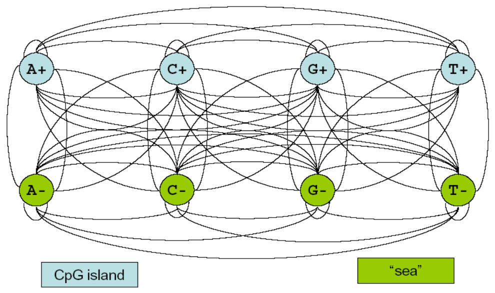
- We can use a sliding window of size 100, for example, around each nucleotide in the sequence and use the previous table to score the log-odds. **CpG islands** would stand out with positive values.

Sliding Window Size

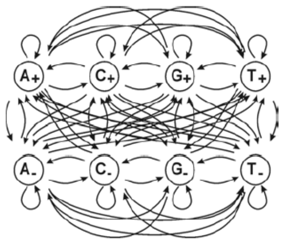


- How do we determine the window size? CpG islands are of variable lengths and might have sharp boundaries.
- A better approach is to build an **HMM** that combines both models.

The Island and the Sea

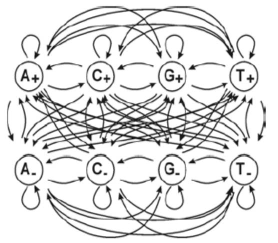


An HMM for CpG Islands (I)



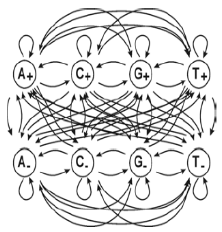
There are 8 states, one for each nucleotide in a **CpG island** (**+**), and one for each nucleotide not in a **CpG island** (**-**).

An HMM for CpG Islands (II)



There two states for each output symbol.
Example: "T" is recognized or generated by T+ or T-.
Within each group of states, the group has the same behavior as the original **Markov Model**.

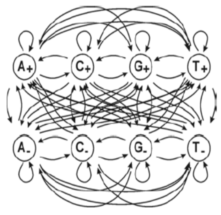
An HMM for CpG Islands (III)

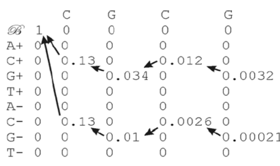


Position i:	Position i+1: C+	G+	C-	G-
C+	0.37	0.27	small	small
G+	0.34	0.38	small	small
C-	small	small	0.3	0.08
G-	small	small	0.25	0.3

Assume the transitions from a (+) nucleotide to a (-) are small. And transitions from (-) nucleotides to (+) are also small.

The Two Paths with CGCG





		C	G	C	G
𝓑	1	0	0	0	0
A+	0	0	0	0	0
C+	0	0.13	0	0.012	0
G+	0	0	0.034	0	0.0032
T+	0	0	0	0	0
A-	0	0	0	0	0
C-	0	0.13	0	0.0026	0
G-	0	0	0.01	0	0.00021
T-	0	0	0	0	0

Position i:	Position i+1: C+	G+	C-	G-
C+	0.37	0.27	small	small
G+	0.34	0.38	small	small
C-	small	small	0.3	0.08
G-	small	small	0.25	0.3

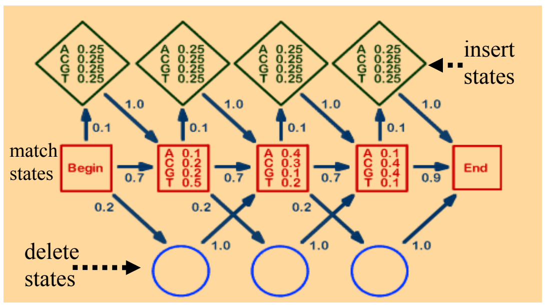
Switching between '+' & '-' States

- The maximum scoring path receives a score of 0.0032.
- The most likely state path is found to be C+G+C+G+.
- Given a much longer sequence, the derived optimal path will switch between the **CpG** and **non-CpG** states.

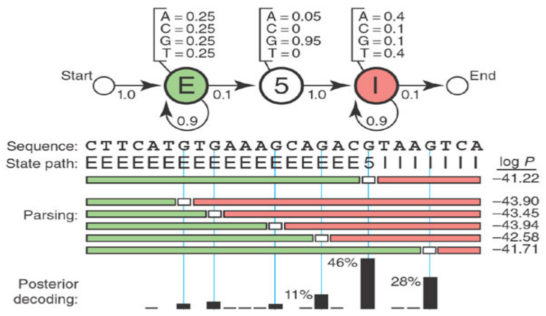
Applications of HMMs

- Generating **multiple sequence alignments**
- **Modeling Protein Family**
  - discriminate between sequences that belong to a particular family or contain a particular domain vs. the ones that do not.
- **Study the model** directly
  - the model may reveal something about the common structure of proteins within a family.
- **Gene prediction**

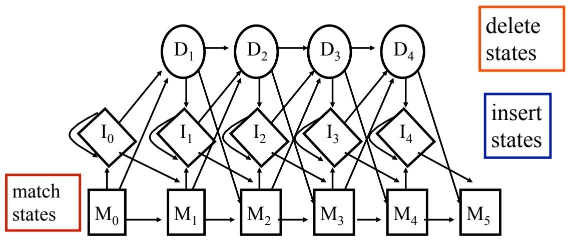
Recognizing TAG



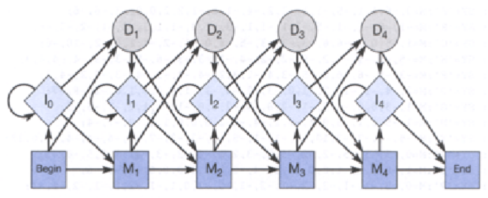
Eddy's Toy Model


A = 0.25 C = 0.25 G = 0.25 T = 0.25
A = 0.05 C = 0 G = 0.95 T = 0
A = 0.4 C = 0.1 G = 0.1 T = 0.4
Start
End
E
5
I
1.0
0.1
0.9
Sequence: CTTCATGTGAAAGCAGACGTAAGTCA
State path: EEEEEEEEEEEEEEEEEE5IIIIIII
log P
−41.22
Parsing:
−43.90
−43.45
−43.94
−42.58
−41.71
Posterior decoding:
46%
11%
28%

HMM for Protein Family



HMM: Begin and End States



The general model with **Begin** and **End** states

Family of Sequences

- If the emission probabilities for the match and insert states are uniform over the 20 amino acids, the model will produce random sequences.
- If each state emits one amino acid only, and transition probabilities from one match state to the next are one, then the model will produce the same sequence.
- Somewhere between the two extreme cases we can set the parameters to obtain a **family of sequences** (sequences that are similar).

The Goal

- Find a **model**, in other words, a model length, and parameters, that accurately describes a family of proteins.
  - The model will assign high probabilities to proteins in the family of sequences that it is designed for.

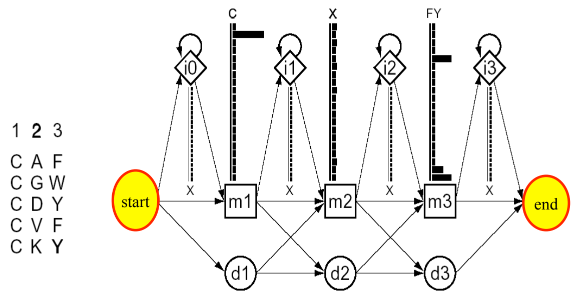
Profile Hidden Markov Model

- Allowing gap penalties and substitutions probabilities to vary along the sequences reflects biological reality better.
- Alignments of related proteins have regions of higher conservation, called **functional domains** and regions of lower conservation.
- **Functional domains** have resisted to change indicating that they serve some critical function.

Estimating the Parameters

- In the HMM model of a protein family, the transition from:
  - a match state to an insert state corresponds to a gap open penalty
  - an insert state to itself corresponds to the gap extension penalty
- All applications of the HMM model start with training or estimating the parameters of the model using a set of training sequences chosen from a protein family.

Profile HMM From MSA



Regular Expressions for MSA

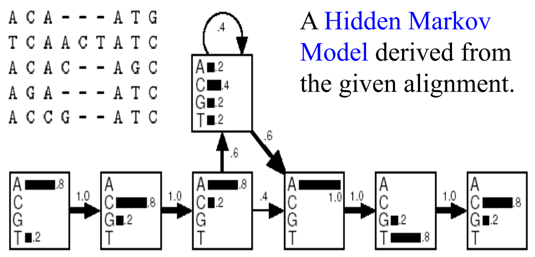
```
A C A - - - A T G
T C A A C T A T C
A C A C - - A G C
A G A - - - A T C
A C C G - - A T C
```

The given DNA motif can be represented by a regular expression:

[AT][CG][AC][ACTG]*A[TG][GC]

Is this a good representation?
The expression does not distinguish between:
TGCT - - AGG → highly implausible sequence ♪
ACAC- - ATC → highly plausible sequence

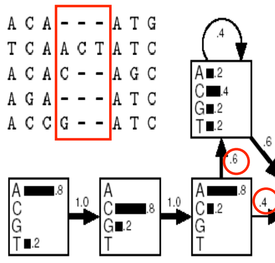
Example: HMM for MSA (I)



```
A C A - - - A T G
T C A A C T A T C
A C A C - - A G C
A G A - - - A T C
A C C G - - A T C
```

A Hidden Markov Model derived from the given alignment.

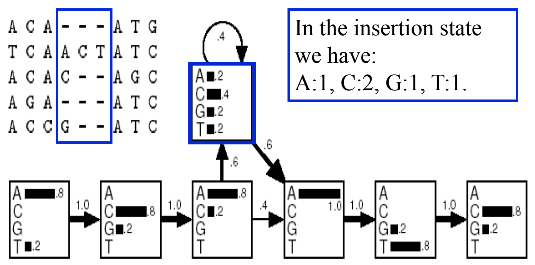
Example: HMM for MSA (II)



```
A C A - - - A T G
T C A A C T A T C
A C A C - - A G C
A G A - - - A T C
A C C G - - A T C
```

Sequences 2, 3 and 5 have "insertions" of varying lengths.
So 3 out of 5 sequences have "insertions".

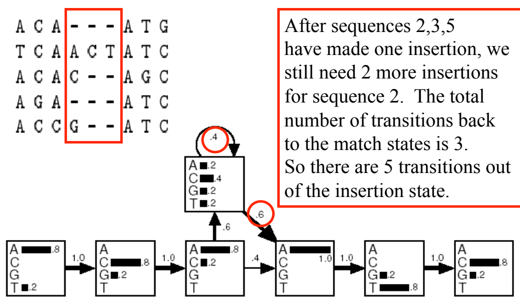
Example: HMM for MSA (III)



```
A C A - - - A T G
T C A A C T A T C
A C A C - - A G C
A G A - - - A T C
A C C G - - A T C
```

In the insertion state we have:
A:1, C:2, G:1, T:1.

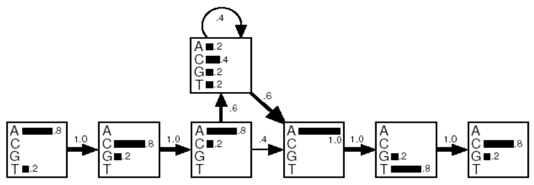
Example: HMM for MSA (IV)



```
A C A - - - A T G
T C A A C T A T C
A C A C - - A G C
A G A - - - A T C
A C C G - - A T C
```

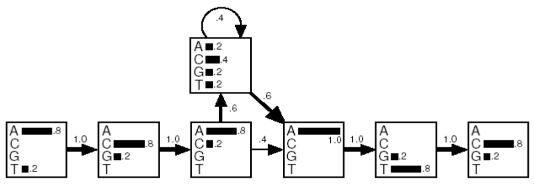
After sequences 2,3,5 have made one insertion, we still need 2 more insertions for sequence 2. The total number of transitions back to the match states is 3. So there are 5 transitions out of the insertion state.

Computing Probability of Path



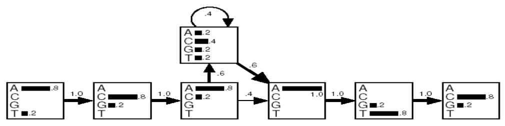
P(ACACATC)
= 0.8 * 1.0 * 0.8 * 1.0 * 0.8 * 0.6 * 0.4 * 0.6 * 1.0 * 1.0 * 0.8 * 1.0 * 0.8
= 0.04718592

Computing Probability of Path (II)



P(TCAACTATC)
= **0.2** * 1.0 * **0.8** * 1.0 * **0.8** * 0.6 * **0.2** * 0.4 * **0.4** * 0.4 * **0.2** * 0.6 * **1.0** * 1.0 * **0.8** * 1.0 * **0.8**
= 0.000075497472

HMM: Computing Log Odds



Using Log Odds of sequence S of length L

= log [P(S) / (0.25)^L]

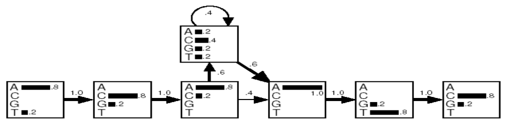
= log P(S) – L log (0.25)

Example: Log Odds of ACACATC

= log (P(ACACATC)) – 7 * log (0.25)

≅ 6.7 [natural logarithm]

Log Odd Scores of Sequences



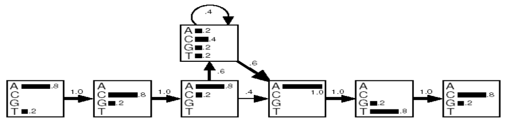
	Sequence	Probability ×100	Log odds
Consensus	A C A C - - A T C	4.7	6.7
Original sequences	A C A - - - A T G	3.3	4.9
	T C A A C T A T C	0.0075	3.0
	A C A C - - A G C	1.2	5.3
	A G A - - - A T C	3.3	4.9
	A C C G - - A T C	0.59	4.6
Exceptional	T G C T - - A G G	0.0023	-0.97

Log Odd Scores of Sequences

	Sequence	Probability ×100	Log odds
Consensus	A C A C - - A T C	4.7	6.7
Original sequences	A C A - - - A T G	3.3	4.9
	T C A A C T A T C	0.0075	3.0
	A C A C - - A G C	1.2	5.3
	A G A - - - A T C	3.3	4.9
	A C C G - - A T C	0.59	4.6
Exceptional	T G C T - - A G G	0.0023	-0.97

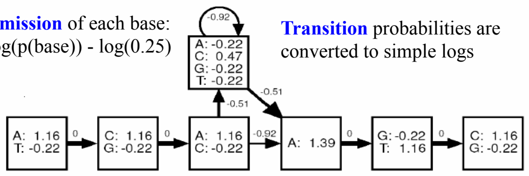
A sequence that fits the motif very well has a high log-odds score.
A sequence that fits the null hypothesis better has a negative log-odds score.

HMM with Log Odds of Each Base

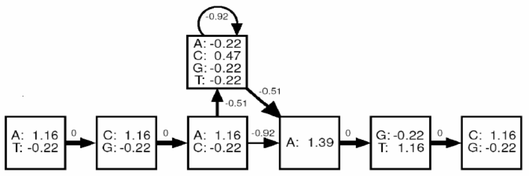


Emission of each base:
log(p(base)) - log(0.25)

Transition probabilities are converted to simple logs



Log Odds Score of a Sequence

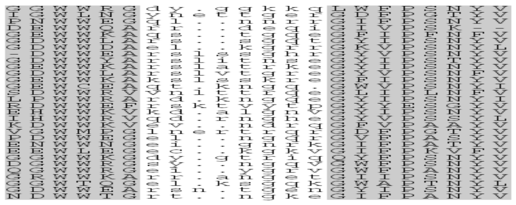


Log Odds score of ACACATC

= 1.16 + 0 + 1.16 + 0 + 1.16 – 0.51 + 0.47 – 0.51 + 1.39 + 0 + 1.16 + 0 + 1.16

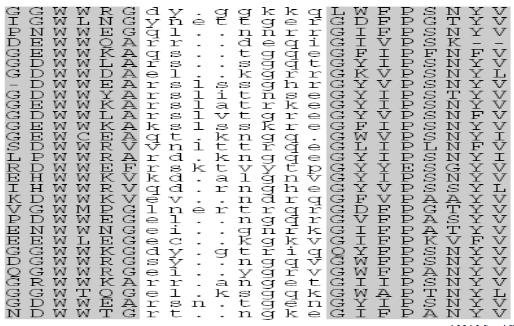
= 6.64

SH3 Domain Example (I)

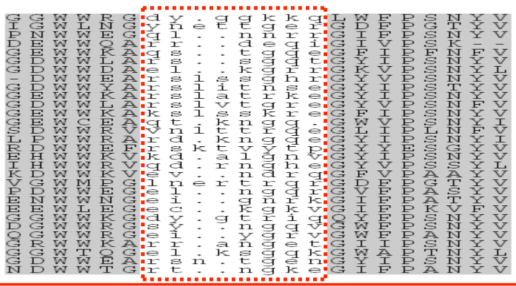


An alignment of 30 short amino acids chopped out of an alignment of an SH3 domain. The shaded areas are the most conserved and were chosen to be represented by the main (match) states and the unshaded area with lower-case letters was chosen to be represented by an insert state. [Kro98]

SH3 Domain Example (II)

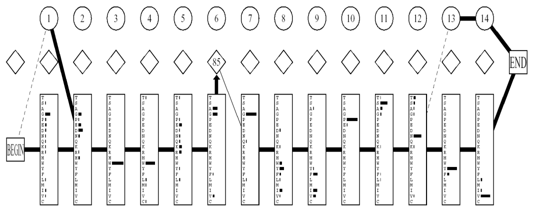


SH3 Domain Example (III)



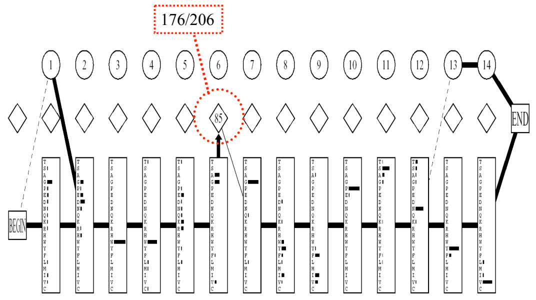
The insert state represents highly variable regions of the alignment

SH3 Domain Example (IV)

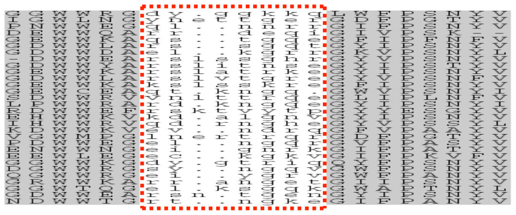


A profile HMM made from the alignment. Transition lines with no arrow head are from left to right. Transitions with probability zero are not shown. Those with very small probability are shown as dashed lines. Transition from an insert state to itself are not shown; instead the probability multiplied by 100 is shown in the diamond. The numbers in the circular delete states are just position numbers.

SH3 Domain Example (V)



SH3 Domain Example (VI)



After all 30 sequences have made one insertion each, there are 176 more insertions (number of amino acids in columns 2,3,4,5,6,7,8 of the insert state) and there are 30 transitions back to the match states. So there are a total of 176 + 30 transitions out the insert state. P(self-loop) = 176/206 and P(back to match) = 30/206.

Markov Model Assumptions (I)

- A set Q of N states, denoted by 1,2,…,N
- An observable sequence, O:

$$o_1, o_2, \dots, o_t, \dots, o_T$$

- An unobservable sequence, q:

$$q_1, q_2, \dots, q_t, \dots, q_T$$

- First order Markov model:

$$P(q_t = j \mid q_{t-1} = i, q_{t-2} = k, \dots) = P(q_t = j \mid q_{t-1} = i)$$

Markov Model Assumptions (II)

- An initial probability distribution:

$$\pi_i = P(q_1 = i) \qquad 1 \le i \le N$$

where $\sum_{i=1}^{N} \pi_i = 1$

- Stationary condition:

$$P(q_t = j \mid q_{t-1} = i) = P(q_{t+l} = j \mid q_{t+l-1} = i)$$

State Transition Probabilities

State transition probability matrix:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{iN} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{Nj} & \cdots & a_{NN} \end{bmatrix}$$

where: $a_{ij} = P(q_t = j \mid q_{t-1} = i) \qquad 1 \le i, j \le N$

$a_{ij} \ge 0, \quad \forall i, j$

$\sum_{j=1}^{N} a_{ij} = 1, \quad \forall i$

Hidden Markov Model

- N: the number of hidden states
  A set of states: $Q = \{1, 2, \ldots, N\}$
- M: the number of symbols
  A set of symbols: $V = \{1, 2, \ldots, M\}$
- A: the state-transition probability matrix
  $a_{i,j} = P(q_{t+1} = j \mid q_t = i) \qquad 1 \le i, j \le N$
- B (or b): Emission probability distribution; *k* is a symbol from V:
  $B_j(k) = P(o_t = k \mid q_t = j) \qquad 1 \le j \le M$
- The initial state distribution π:
  $\pi_i = P(q_1 = i) \qquad 1 \le i \le N$

  The entire model λ, is given by: $\lambda = (A, B, \pi)$

Three Basic Questions

1. EVALUATION – given observation $O=(o_1, o_2, \ldots, o_T)$ and model $\lambda = (A, B, \pi)$, efficiently compute: $P(O \mid \lambda)$.
   - Given two models λ and λ' , this can be used to choose the better one.
   Use: **Forward Algorithm** or **Backward Algorithm**
2. DECODING - given observation $O=(o_1, o_2, \ldots, o_T)$ and model λ find the optimal state sequence $q=(q_1, q_2, \ldots, q_T)$.
   - Optimality criterion has to be decided (e.g. maximum likelihood)
   Use: **Viterbi Algorithm**
3. LEARNING – given $O=(o_1, o_2, \ldots, o_T)$, estimate model parameters $\lambda = (A, B, \pi)$ that maximize $P(O \mid \lambda)$.
   Use: **EM and Baum-Welch Algorithms**

Doubly Stochastic Process

According to Rabiner and Juang:

A **Hidden Markov Model** is a doubly stochastic process with an underlying stochastic process which is not observable (it is hidden), but can be only observed through another set of stochastic processes that produce the sequence of observed symbols.

(IEEE ASSP, January 1986)

HMM and Logarithms

- In a Hidden Markov Model there is not a one to one correspondence between the states and the symbols as is the case with Markov Chains.
- Extensive multiplication operations with probabilities often result in underflows.
  – Use logarithms: products become sums.

Scoring a Sequence

- All sequences will have a path through the HMM.
- For most sequences (except very short ones) there are a huge number of paths through the model, most of which will have very low probability values.
- For a given observed sequence, we can approximate the total probability by the probability of the most likely path.
  - **Viterbi**: method for finding the most likely path.

Viterbi: A Summary

- Similar to Dynamic Programming already studied.
- Make a matrix with rows for sequence elements and columns for states in the model.
- Work row by row, calculating the probability for each state to have emitted that element and putting that probability in a cell.
  - When there are multiple paths, select the highest probability and store the selected path.
- Current row uses results of previous row.
- Highest entry in the last row gives best total path through back tracking.

Three Basic Questions (I)

1. **EVALUATION** – given observation $O=(o_1, o_2, \dots, o_T)$ and model $\lambda = (A, B, \pi)$, efficiently compute $P(O \mid \lambda)$.
   - Hidden states complicate the evaluation.
   - Given two models λ and λ', this can be used to choose the better one.
2. **DECODING** - given observation $O=(o_1, o_2, \dots, o_T)$ and model λ find the optimal state sequence $q=(q_1, q_2, \dots, q_T)$.
   - Optimality criterion has to be decided (e.g. maximum likelihood)
   - "Explanation" of the data.
3. **LEARNING** – given $O=(o_1, o_2, \dots, o_T)$, estimate model parameters $\lambda = (A, B, \pi)$ that maximize $P(O \mid \lambda)$.

Three Basic Questions (II)

- **The Evaluation Problem**
  - Given the observation sequence O and the model λ, how do we efficiently compute P(O| λ), the probability of the observation sequence, given the model?
- **The Decoding Problem**
  - Given the observation sequence O and the model λ, find the optimal state sequence associated O.
    Viterbi Algorithm finds the single best sequence q for the given observation sequence O.
- **The Learning Problem**
  - How can we adjust the model parameters to maximize the joint probability (likelihood)?

Three Basic Questions (III)

- **The Evaluation Problem**
  - Given the observation sequence O and the model λ, how do we efficiently compute P(O| λ), the probability of the observation sequence, given the model?
    Use: **Forward Algorithm** or **Backward Algorithm**.
- **The Decoding Problem**
  - Given the observation sequence O and the model λ, find the optimal state sequence associated with O. Viterbi Algorithm finds the single best sequence q for the given observation sequence O.
    Use: **Viterbi Algorithm**.
- **The Learning Problem**
  - How can we adjust the model parameter to maximize the joint probability (likelihood)?
    Use: **EM and Baum-Welch Algorithms**.

Solution to Problem One (I)

Problem: Compute $P(o_1, o_2, \dots, o_T / \lambda)$

$$P(O \mid \lambda) = \sum_q P(O, q \mid \lambda)$$ the summation is over all paths $q = (q_1, q_2, \dots, q_T)$ that give O.

But: $P(O, q \mid \lambda) = P(O \mid q, \lambda) P(q \mid \lambda)$

$$P(O \mid q, \lambda) = \prod_{t=1}^{T} P(o_t \mid q_t, \lambda)$$

We assume that the observations are independent

$$= b_{q_1}(o_1) b_{q_2}(o_2) \dots b_{q_T}(o_T)$$

Solution to Problem One (II)

We also have: $P(q \mid \lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \dots a_{q_{T-1} q_T}$

By replacing in: $P(O \mid \lambda) = \sum_q P(O, q \mid \lambda)$

we have:

$$P(O \mid \lambda) = \sum_q \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) a_{q_2 q_3} b_{q_3}(o_3) \dots a_{q_{T-1} q_T} b_{q_T}(o_T)$$

We have: 2T – 1 multiplications and a maximum of N^T state sequences and $O(T)$ calculations.

Complexity: $O(T N^T)$.

Solution to Problem One (III)

Since the complexity is $O(T N^T)$, the brute force evaluation of:

$$P(O \mid \lambda) = \sum_q P(O, q \mid \lambda)$$

by enumerating all paths q that generate O is not practical.

To efficiently compute $P(o_1, o_2, \dots, o_T \mid \lambda)$, use the **Forward Algorithm**.

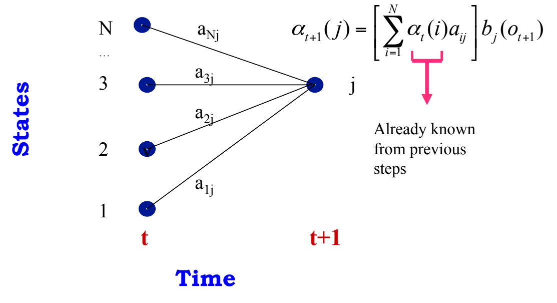
Forward Algorithm

- Define **forward variable** $\alpha_t(i)$ as:

$$\alpha_t(i) = P(o_1, o_2, \dots, o_t, q_t = i \mid \lambda)$$

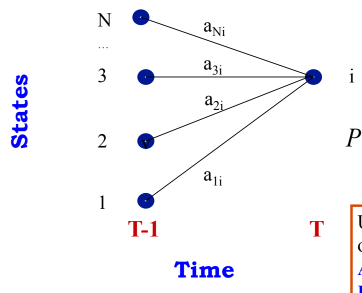
- $\alpha_t(i)$ is the probability of observing the partial sequence $(o_1, o_1, \dots, o_t)$ and landing in state i at stage t (state q_t is i).
- **Recurrence Relation:**
  1. Initialization: $\alpha_1(i) = \pi_i b_i(o_1)$
  2. Induction: $\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1})$
  3. Termination: $P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$
- **Complexity:** $O(N^2 T)$

Forward Procedure: Induction



$$\alpha_{t+1}(j) = \left[\underbrace{\sum_{i=1}^{N} \alpha_t(i) a_{ij}}\right] b_j(o_{t+1})$$

Forward Procedure: Termination



$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Use either **Forward** or **Backward Algorithm** to solve **Problem One**.

Backward Algorithm

- Define **backward variable** $\beta_t(i)$ as:

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \dots, o_T \mid q_t = i, \lambda)$$

$\beta_t(i)$ is the probability of observing the partial sequence $(o_{t+1}, o_{t+2}, \dots, o_T)$ knowing that we land in state i at stage t (in other words, state q_t is i).

Backward: Recurrence Relation

- **Recurrence Relation** of $\beta_t(i)$
  - Initialization: $\beta_T(i) = 1$
  - Induction:

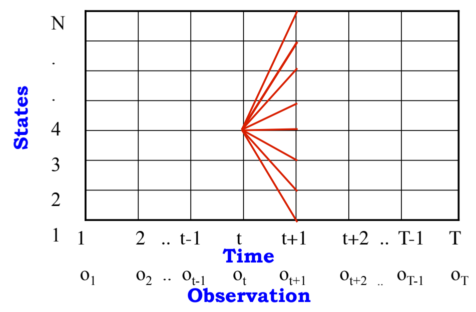
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \qquad 1 \le i \le N, \qquad t = T-1, \ldots, 1$$

  - Termination:

$$P(O \mid \lambda) = \sum_{j=1}^{N} \pi_j b_j(o_1) \beta_1(j)$$

- **Complexity:** $O(N^2 T)$

Backward Algorithm



Backward Algorithm: Remark

- **Backward variable** $\beta_t(i)$ is given by:

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T \mid q_t = i, \lambda)$$

- We note that, unlike the forward variable, here we know in which state the process is at time t (state $q_t = i$).
- The distinction is made to be able to combine the **forward** and **backward** variables to produce a useful result.

Using Forward and Backward (I)

Compute the probability of producing the entire observed sequence, *O*, with the tth symbol produced by state i.

$$P(q_t = i \mid O) = \frac{P(O, q_t = i)}{P(O)}$$ We drop λ for convenience

$$\begin{aligned} P(O, q_t = i) &= P(o_1, o_2, \ldots o_t, o_{t+1}, \ldots o_{T-1}, o_T, q_t = i) \\ &= P(o_{t+1}, \ldots o_{T-1}, o_T, o_1, o_2, \ldots o_t, q_t = i) \\ &= P(o_1, o_2, \ldots o_t, q_t = i) P(o_{t+1}, \ldots o_{T-1}, o_T \mid o_1, o_2, \ldots o_t, q_t = i) \\ &= P(o_1, o_2, \ldots o_t, q_t = i) P(o_{t+1}, \ldots o_{T-1}, o_T \mid q_t = i) \\ &= \alpha_t(i) \beta_t(i) \end{aligned}$$

Using Forward and Backward (II)

$$P(q_t = i \mid O) = \frac{P(O, q_t = i)}{P(O)} = \frac{\alpha_t(i)\beta_t(i)}{P(O)}$$

P(O) can be computed by using either the **forward** or **backward** algorithm.

Or:

$$P(q_t = i \mid O, \lambda) = \frac{P(O, q_t = i, \lambda)}{P(O, \lambda)} = \frac{\alpha_t(i)\beta_t(i)}{P(O, \lambda)}$$

Solution to Problem 2 (I)

- We have to find a state sequence: $q = (q_1, q_2, \ldots, q_T)$, such the probability of occurrence of the observed sequence: $O = (o_1, o_2, \ldots, o_T)$ from the state sequence *q*, is greater than or equal to any other state sequence.
- Find a path $q^* = (q_1^*, q_2^*, \ldots, q_T^*)$ that maximizes the likelihood:

$$P(q_1, q_2, \ldots, q_T \mid O, \lambda)$$

Solution to Problem 2 (II)

- The **Viterbi algorithm** can be used to solve this problem.
- It is a modified forward algorithm.
- Instead of taking the sum of all possible paths that end up in a destination state, the **Viterbi algorithm** picks and remembers the best path.

Solution to Problem 3 (III)

Use **Dynamic Programming**

- Define

$$\delta_t(i) = \max_{q_1,q_2,\ldots q_t} P(q_1,q_2,\ldots q_t = i, o_1,o_2 \ldots o_t \mid \lambda)$$

$\delta_t(i)$ is the highest probability path ending in state *i* at step *t* (time *t*).

- By induction we have:

$$\delta_{t+1}(j) = \max_i [\delta_t(i)a_{ij}] \cdot b_j(o_{t+1})$$

Viterbi Algorithm (I)

- **Initialization**:

$$\delta_1(i) = \pi_i b_i(o_1), \quad 1 \le i \le N$$
$$\psi_1(i) = 0$$

- **Recursion**:

$$\delta_t(j) = \max_{1\le i\le N}[\delta_{t-1}(i)a_{ij}]b_j(o_t)$$
$$\psi_t(j) = \arg\max_{1\le i\le N}[\delta_{t-1}(i)a_{ij}]$$
$$2 \le t \le T, \quad 1 \le j \le N$$

Viterbi Algorithm (II)

- **Termination:**

$$P_T^* = \max_{1\le i\le N}[\delta_T(i)]$$
$$q_T^* = \arg\max_{1\le i\le N}[\delta_T(i)]$$

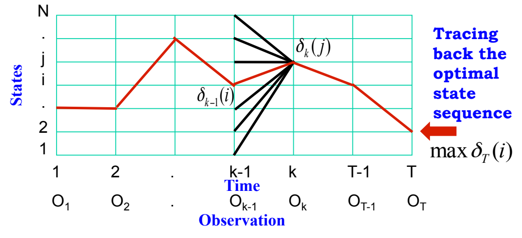
where

$$P_T^* = P(q_1,q_2,\ldots,q_T \mid O,\lambda)$$

- A maximum likelihood path is given by: $q^* = (q_1^*, q_2^*, \ldots, q_T^*)$, where

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \ldots, 1$$

Viterbi Algorithm (III)



Solution to Problem 3

- Estimate $\lambda = (A, B, \pi)$ to maximize $P(O \mid \lambda)$
- No analytic methods exist because of complexity – Use an iterative solution.
- **Expectation Maximization**: the **EM algorithm**
  1. Let initial model be λ_0
  2. Compute new λ based on λ_0 and observation O.
  3. If $\log P(O \mid \lambda) - \log P(O \mid \lambda_0) < DELTA$ stop
  4. Else set $\lambda_0 \longleftarrow \lambda$ and go to step 2

EM Special Case: Baum-Welch

- The **Expectation Maximization Algorithm** is a very powerful general algorithm for probabilistic parameter estimation.
- The **Baum-Welch Algorithm** is a special case of the **Expectation Maximization Algorithm**.

Parameter Estimation for HMMs

There are two parts for specifying a **Hidden Markov Model**:

1. Design of the **structure** (more of an art)
   - Determining the states
   - Determining the connections of the states
2. Assignment of **parameter values** (a well-developed theory exists)
   - Determining the transition and emission probabilities.

Assignment of Parameter Values

- There are two cases to consider when assigning **parameter values** to HMMs:
  - Estimation when the state sequence is known
    - **Example**: Location of CpG islands are already known
  - Estimation when the state sequences are unknown.

Estimation with Known State Paths

Estimation of the parameters is straightforward when sequence paths are known.

- Count the number of times a particular **transition** (denoted by **A**) or **emission** (denoted by **B**) is used in the training set
- The **maximum likelihood estimations** are:

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}} \qquad b_k(d) = \frac{B_k(d)}{\sum_{d'} B_k(d')}$$

The Dangers of Overfitting

When estimating parameters, especially from a limited amount of data, there is a danger of **overfitting**: the model becomes very well adapted to the training data and does not generalize well to testing data (new data).

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}} \qquad b_k(d) = \frac{B_k(d)}{\sum_{d'} B_k(d')}$$

Pseudocounts to the Rescue

To avoid **overfitting**, add predetermined pseudocounts r_{kl} & $r_k(d)$ to the numerators of the transition estimators:

A_{kl} is the number of **transitions** k to l in the training data + r_{kl}

$B_k(d)$ is the number of **emissions** of d from state k in the training data + $r_k(d)$

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}} \qquad b_k(d) = \frac{B_k(d)}{\sum_{d'} B_{k'}(d')}$$

Estimation if Paths are Unknown

- When paths are unknown for training sequences, we have no direct closed-form equation for the **estimated parameter values**.
- **Iterative procedures** are used.
- The **Baum-Welch** algorithm (special case of the **EM** algorithm) has become the standard method when paths are unknown.

The Two Steps of Baum-Welch

- The **Baum-Welch Algorithm** is based on the following observation:
  - If we knew the **paths**, we could compute **transition** and **emission** probabilities
  - If we knew the **transition** and **emission** probabilities, we could compute the **paths** (for example: the most probable path)
- The algorithm alternates between the two.

Baum-Welch Iterative Process

- The **Baum-Welch Algorithm** is basically an iterative process that alternates between the following two steps:
  - Estimate A_{kl} and $B_k(d)$ by considering probable paths for the training sequence using the current values of a_{kl} and $b_k(d)$ [Expectation]
  - Derive new values by using above values in: [Maximization]

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}} \qquad b_k(d) = \frac{B_k(d)}{\sum_{d'} B_k(d')}$$

- Iterate until some stopping criterion is reached.

Baum-Welch at Work (I)

- The probability that a_{kl} is used at position t in the observed sequence $O=(o_1, o_2,\ldots,o_T)$ is given by:

$$P(q_t = k, q_{t+1} = l \mid O,\lambda) = \frac{\alpha_t(k)a_{kl}b_l(o_{t+1})\beta_{t+1}(l)}{P(O,\lambda)}$$

- Then the expected number of times that a_{kl} is used is obtained by summing over all positions and over all **training sequences**:

$$A_{kl} = \sum_j \frac{1}{P(o^j)} \sum_t \alpha_t^j(k)a_{kl}b_l(o_{t+1}^j)\beta_{t+1}^j(l)$$

Baum-Welch at Work (II)

$$A_{kl} = \sum_j \frac{1}{P(o^j)} \sum_t \alpha_t^j(k)a_{kl}b_l(o_{t+1}^j)\beta_{t+1}^j(l)$$

$\alpha_t^j(k)$ is the forward variable for training sequence j

$\beta_t^j(k)$ is the backward variable for training sequence j.

- Similarly, the expected number of times that symbol d is emitted from state k in all the sequences is given by:

$$B_k(d) = \sum_j \frac{1}{P(o^j)} \sum_{\{t \mid o_t^j = d\}} \alpha_t^j(k)\beta_t^j(k)$$

The inner sum is only over positions t for which the emitted symbol is *d*.

Baum-Welch Iteration

- Use the newly computed expectation values: A_{kl} and $B_k(d)$ to calculate the new model transition and emission parameters :

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}} \qquad b_k(d) = \frac{B_k(d)}{\sum_{d'} B_k(d')}$$

- We then compute again A_{kl} and $B_k(d)$ based on the new parameters and iterate once more.

Baum-Welch Algorithm

- **Initialization**:
  - Pick arbitrary model parameters
- **Recurrence**:
  - Set all the A and B variables to their pseudocount values r (or zero)
  - For each sequence j = 1,…,n
    - Use forward algorithm to compute $\alpha_i^j(k)$
    - Use backward algorithm to compute $\beta_i^j(k)$
    - Add the contribution of sequence j to A and B
  - Compute the new model parameters
  - Compute the new log likelihood of the model

Termination Step

- **Termination**:

  Stop when the change in the log likelihood is less than some predefined threshold or the maximum number of iterations is reached
- It can be shown that the overall log likelihood is increased by the iteration and that the process converges to a local maximum.
  - One of the challenges of designing HMMs:
    - How good is that local maximum?