CSE8803/CX4803 Machine Learning in Computational Biology

Lecture 4: Gene/Motif finding using HMMs I

Xiuwei Zhang

School of Computational Science and Engineering

Paper presentation logistics

- Three steps: form teams → select dates → select papers
- Form teams
  - 22 groups of 2 students; 11 groups of 3 students
  - **Submit your team information to Canvas->Quizzes by 1/28 Friday (no grace period)**
  - We may adjust the teams (randomly); Teams can also slightly change after midterm withdraw
  - Teams are finalized by 1/30 or 1/31
- List of papers are available to students from Monday 1/24
- Date selection (bidding) submission by 2/2 (Form released on 1/31)

2/21/2022	Phase 1 presentations	Student presentation 1-3
2/23/2022		Student presentation 4-6
2/28/2022	Learning from structure data	RNA structure prediction
3/2/2022		Deep learning for structures (protein structure prediction)
3/7/2022	Phase 2 presentations	Student presentation 7-9
3/9/2022	Learning from network data	Network basics & traditinal ML for graphs
3/14/2022		Network embeddings
3/16/2022	Phase 3 presentations	Student presentation 10-12
3/21/2022	Spring break	No class (Spring Break)
3/23/2022		No class (Spring Break)
3/28/2022	Learning from network data	Graphical Models
3/30/2022		Deep learning for networks (graph neural networks)
4/4/2022	Phase 4 presentations	Student presentation 13-15
4/6/2022		Student presentation 15-18
4/11/2022		Student presentation 18-21
4/13/2022		Student presentation 22-24
4/18/2022		Student presentation 25-27
4/20/2022		Student presentation 28-30
4/25/2022		Student presentation 31-33

- 4 phases of presentations
- 4 sets of papers
- Students in the Phase 1 can only choose papers from Set 1 but have priority to choose papers from Set 1
- Instructions and paper list is at https://docs.google.com/document/d/1RJDWddTV3hqngIS6YGzSXGpKCIYgjL5k_GtLarOU54s/edit?usp=sharing

Multiple sequence alignment: Star Alignment

Idea: Build a multiple sequence alignment up from pairwise alignments.

Start with an alignment between S_c and some other sequence:

```
SC YFPHFDLSHGSAQVKAHGKKVGDALTLAVGHLDDLPGAL
S1 YFPHFDLSHG-AQVKG--KKVADALTNAVAHVDDMPNAL
```

Add 3rd sequence, say S2, and use the SC - S2 alignment as a guide, adding spaces into the MSA as needed.

SC - S2 alignment:

```
SC YFPHF-DLS-----HGSAQVKAHGKKVGDALTLAVGHL----DDLPGAL
S2 FFPKFKGLTTADQLKKSADVRWHAERII----NAVNDAVASMDDTEKMS
```

New {SC, S1, S2} alignment (*carry all gaps from pairwise alignments*):

```
SC          YFPHF-DLS-----HGSAQVKAHGKKVGDALTLAVGHL----DDLPGAL
S1          YFPHF-DLS-----HG-AQVKG--KKVADALTNAVAHV----DDMPNAL
S2 FFPKFKGLTTADQLKKSADVRWHAERII----NAVNDAVASMDDTEKMS
```

Continue with S3, S4, ...

Other progressive alignment strategy

First align the most similar sequences

How do we represent an alignment such that we can align a sequence to an alignment, or align two alignments?

Profiles

- Another way to summarize an MSA:

```
S1    ACG-TT-GA
S2    ATC-GTCGA
S3    ACGCGA-CC
S4    ACGCGT-TA
```

Column in the alignment

Character

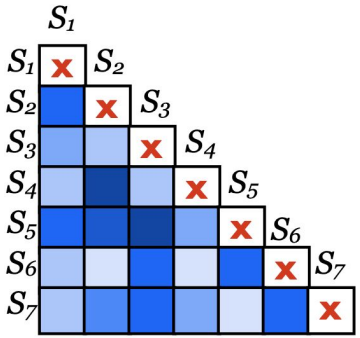
	1	2	3	4	5	6	7	8	9
A	1	0	0	0	0	0.25	0	0	0.75
C	0	0.75	0.25	0.5	0	0	0.25	0.25	0.25
G	0	0	0.75	0	0.75	0	0	0.5	0
T	0	0.25	0	0	0.25	0.75	0	0.25	0
-	0	0	0	0.5	0	0	0.75	0	0

Call this profile matrix R

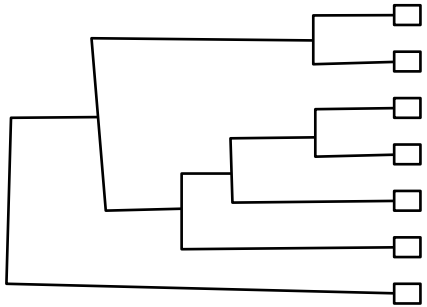
Fraction of time given column had the given character

CLUSTLW

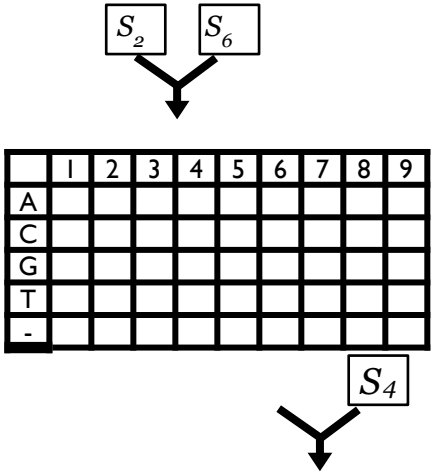
- CLUSTLW is a widely used, "classical" heuristic multiple aligner.
- Not the fastest, not the most accurate, but pretty good.
- Large # of heuristic tricks included in the software, but basic idea is straightforward:



(1): Build pairwise distance matrix



(2): Build guide tree



(3): Align sequences / **sets of sequences** from the most similar to least similar

Profile-based Alignment

gap in profile introduced
to better fit sequence

R =

	1	2	3	4		5	6	7	8	9
A	1	0	0	0	-	0	0.25	0	0	0.75
C	0	0.75	0.25	0.5	-	0	0	0.25	0.25	0.25
G	0	0	0.75	0	-	0.75	0	0	0.5	0
T	0	0.25	0	0	-	0.25	0.75	0	0.25	0
-	0	0	0	0.5	-	0	0	0.75	0	0

A C G - A G A C G A

Score of matching character x with column j of the profile:

$$P(x, j) = \sum_{c \in \Sigma} \text{sim}(x, c) \times R[c, j]$$

sim(x,c) = how similar character x is to character c.

$$A[i, j] = \max \begin{cases} A[i-1, j-1] + P(x_i, j) & \text{align } x_i \text{ to column } j \\ A[i-1, j] + \text{gap} & \text{introduce gap into profile} \\ A[i, j-1] + P(\text{“-“}, j) & \text{introduce gap into } x \end{cases}$$

MSA Recap

- Multiple sequence alignments (MSAs) are a fundamental tool. They help reveal subtle patterns, compute consistent distances between sequences, etc.
- Quality of MSAs often measured using the SP-score: sum of the scores of the pairwise alignments implied by the MSA.
- Same DP idea as pairwise alignment leads to exponentially slow algorithm for MSA for general p.
- 2-approximation obtainable via star alignments.
- MSAs often used to create profiles summarizing a family of sequences.

Further reading

- Durbin, R., Eddy, S. R., Krogh, A. & Mitchison, G. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. (Cambridge University Press, 1998), Chapter 6.
- Feng, D.-F. & Doolittle, R. F. Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. *J. Mol. Evol.* **25**, 351–360 (1987)
- Higgins, D. G., Thompson, J. D. & Gibson, T. J. [22] Using CLUSTAL for multiple sequence alignments. in *Methods in Enzymology* vol. 266 383–402 (Academic Press, 1996).
- Thompson, J. D., Linard, B., Lecompte, O. & Poch, O. A comprehensive benchmark study of multiple sequence alignment methods: current challenges and future perspectives. *PLoS One* **6**, e18093 (2011)
- Notredame, C. Recent progress in multiple sequence alignment: a survey. *Pharmacogenomics* **3**, 131–144 (2002)

Hidden Markov Models

HMMs are a type of *generative models*.

Generative models vs *discriminative models*:

- Generative models model the joint distribution of observed variables X and target variables (eg. labels) Y, and can perform tasks including classifying, and sampling more data
- Discriminative models model the conditional distribution P(Y|X), and are used for classification

HMM, Bayesian network, Markov random fields, Gaussian mixture model, deep models (VAE, etc)

Logistic regression, SVM, decision trees, k-nearest neighbor algorithms, conditional random fields

Flipping a Coin

Fair coin:
Pr(Heads) = 0.5

Biased coin:
Pr(Heads) = 0.75

?

Suppose either a fair or biased coin was used to generate a sequence of heads & tails. But we don't know which type of coin was actual used.

Heads/Tails: ↑ ↑ ↓ ↓ ↓ ↓ ↑ ↑ ↑ ↑ ↓ ↑ ↓ ↑ ↓ ↑ ↓ ↑

Flipping a Coin

Fair coin:
Pr(Heads) = 0.5

Biased coin:
Pr(Heads) = 0.75

?

Suppose either a fair or biased coin was used to generate a sequence of heads & tails. But we don't know which type of coin was actual used.

Heads/Tails: ↑ ↑ ↓ ↓ ↓ ↓ ↑ ↑ ↑ ↑ ↓ ↑ ↓ ↑ ↓ ↑ ↓ ↑

Which coin is more likely to have generated this sequence of observations?

Compute the Probability of the Observed Sequence

Fair coin: Pr(Heads) = 0.5
Biased coin: Pr(Heads) = 0.75

X = ↑ ↑ ↓ ↓ ↓ ↓ ↑

Pr(x | Fair) = 0.5 X 0.5 X 0.5 X 0.5 X 0.5 X 0.5 X 0.5 = 0.0078125

Pr(x | Biased) = 0.75 X 0.75 X 0.25 X 0.25 X 0.25 X 0.25 X 0.75 = 0.00165

The log-odds score:

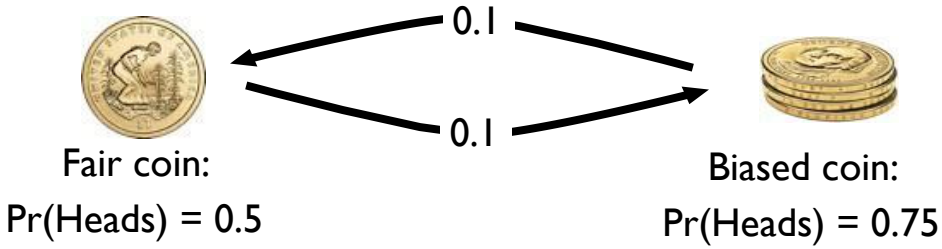
$$\log_2 \frac{\Pr(x \mid \text{Fair})}{\Pr(x \mid \text{Biased})} = \log_2 \frac{0.0078}{0.0016} = 2.245$$

> 0. Hence "Fair" is a better guess.

What if the we switch coins?

Fair coin: Pr(Heads) = 0.5
Biased coin: Pr(Heads) = 0.75

Probability of switching coins = 0.1



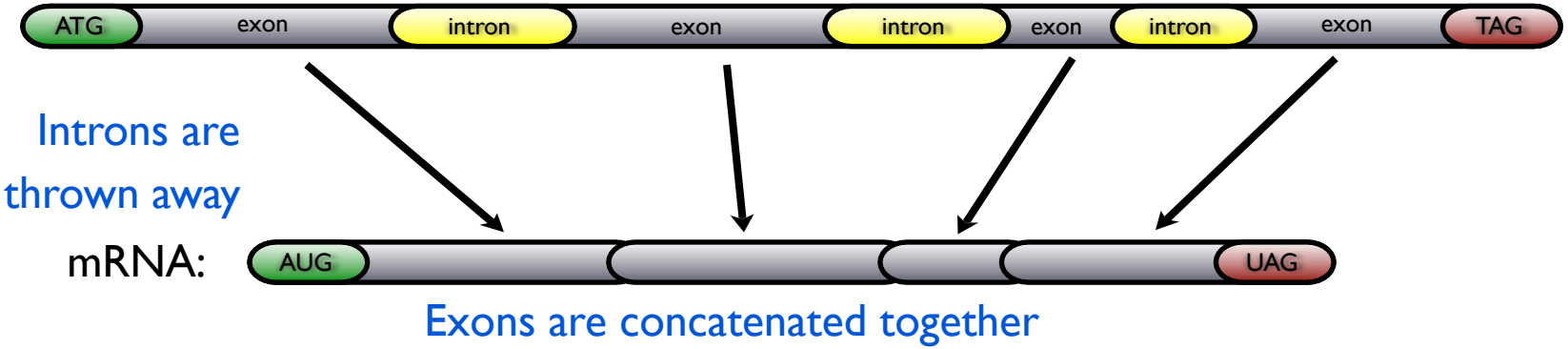
How can we compute the probability of the entire sequence?

How could we guess which coin was more likely **at each position**?

Application in biology

Prokaryotic (bacterial) genes look like this:

Eukaryotic genes usually look like this:



This spliced RNA is what is translated into a protein.

Application in biology

With the flipping coin example:

How likely is it that this sequence was generated by a fair coin?

Which parts were generated by a biased coin?

atg gat ggg agc aga tca gat cag atc agg gac gat aga cga tag tga

With the biological example:

How likely is it that a certain part of a sequence is a gene?

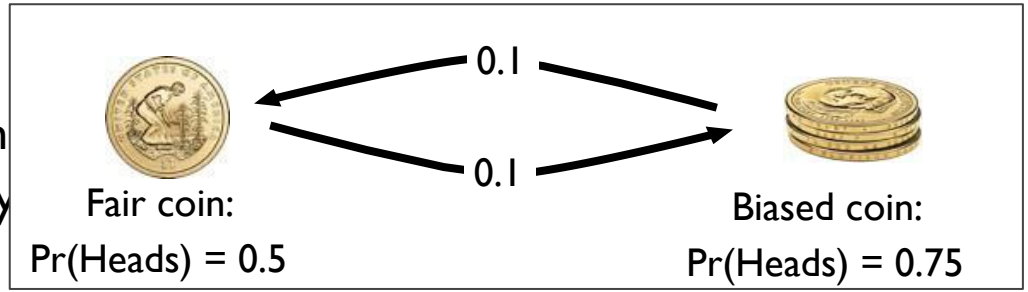
Which parts are the start, middle and end?

Application in biology

With the flipping coin example:

How likely is it that this sequen

Which parts were generated by

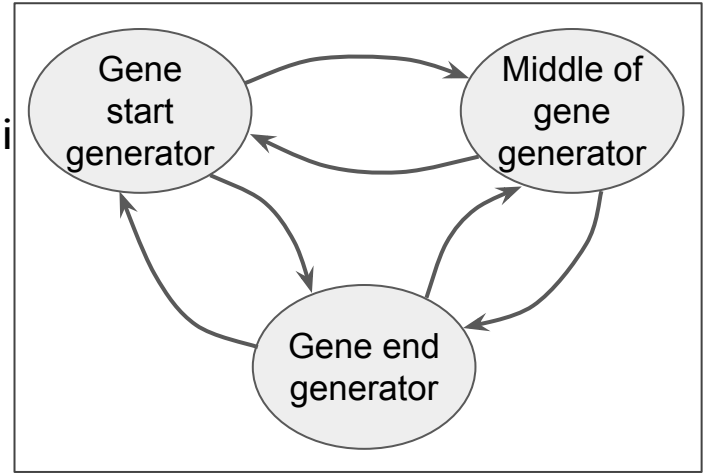


atg gat ggg agc aga tca gat cag atc agg gac gat aga cga tag tga

With the biological example:

How likely is it that a certain part of a sequence i

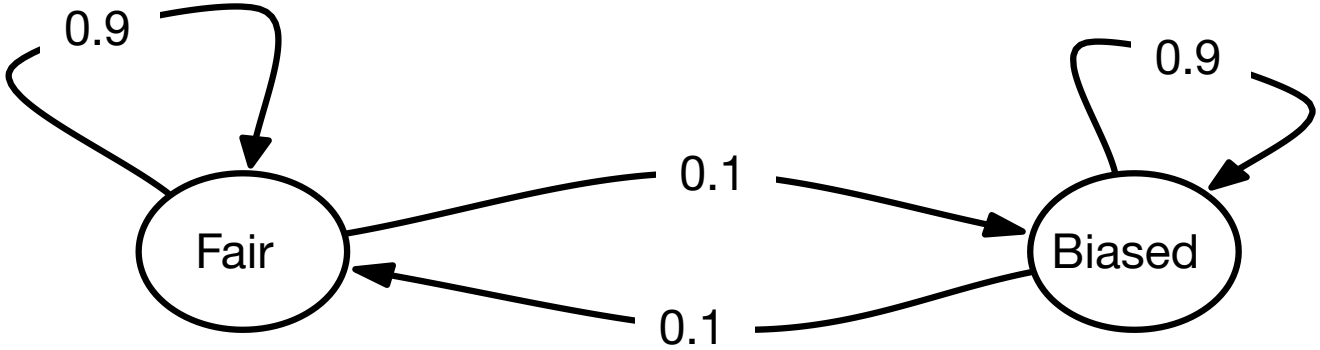
Which parts are the start, middle and end?



Hidden Markov Model (HMM)

Fair coin: Pr(Heads) = 0.5
Biased coin: Pr(Heads) = 0.75

Probability of switching coins = 0.1



	H	T
Fair	0.5	0.5
Biased	0.75	0.25

Formal Definition of a HMM

V = alphabet of symbols, |V|=M

S = set of states, |S|=N

A = an |S| x |S| matrix where entry (i,j) is the probability of moving from state i to state j.

B = a |S| x |V| matrix, where entry (i,k) is the probabili of emitting v_k when in state s_i.

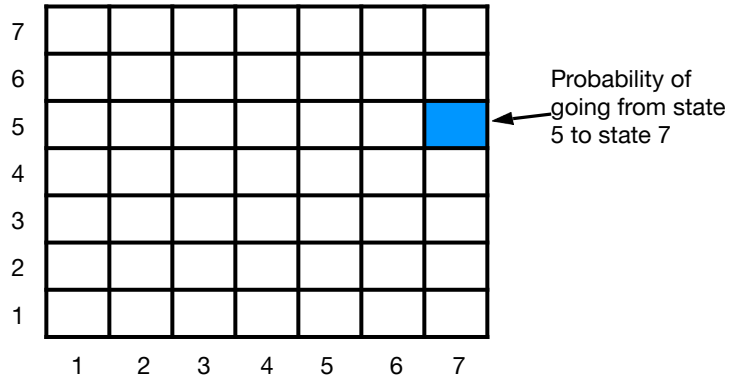
$$\lambda = (\pi, A, B)$$

$$\pi = \{\pi_i\}$$

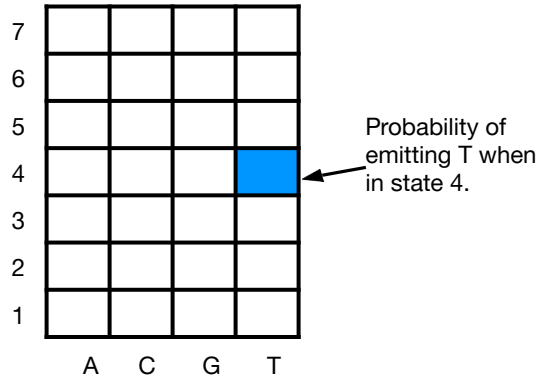
$$A = \{a_{ij}\}$$

$$B = \{b_i(v_k)\}$$

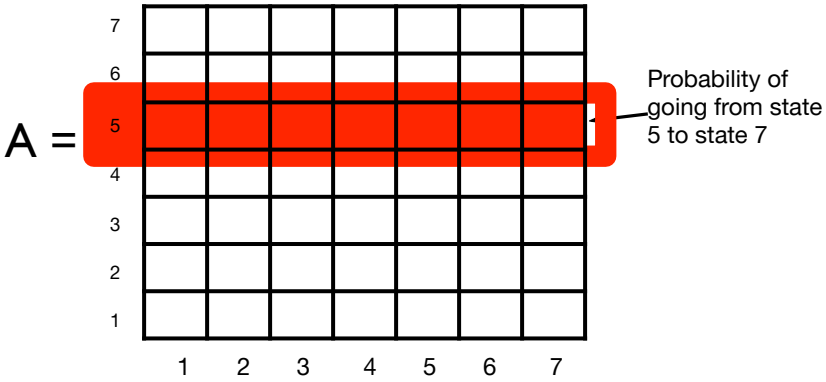
A =

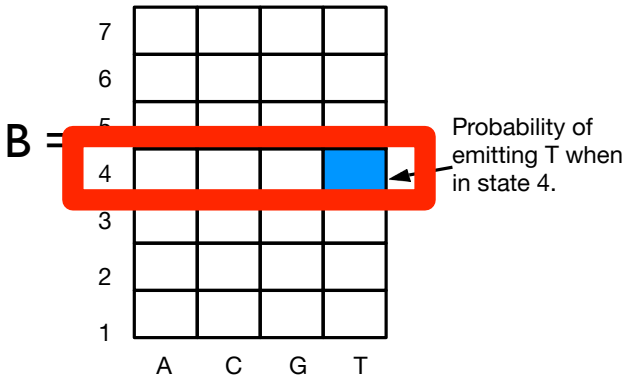


B =



Constraints on A and B





Sum of the # in each row must be 1.

$$\pi = \{\pi_i\}$$

$$A = \{a_{ij}\}$$

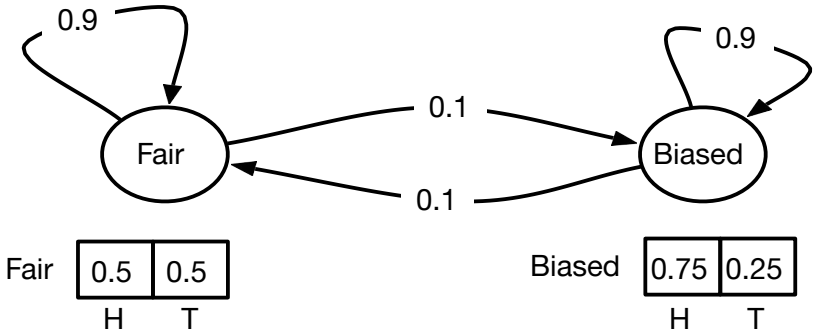
$$B = \{b_i(v_k)\}$$

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

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

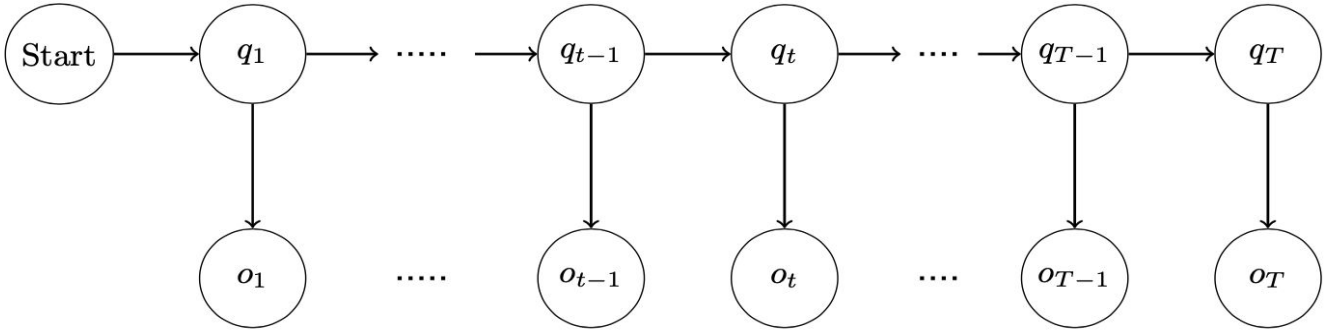
$$\sum_{k=1}^{M} b_i(v_k) = 1, 1 \leq i \leq N$$

Computing Probabilities Given Path



O =	↓	↑	↓	↑	↑	↑	↓	↑	↑	↓
Q =	F	F	F	B	B	B	B	F	F	F
$Pr(o_i \mid q_i)$ =	0.5	0.5	0.5	0.75	0.75	0.75	0.25	0.5	0.5	0.5
$Pr(q_i \rightarrow q_{i+1})$ =	0.9	0.9	0.1	0.9	0.9	0.9	0.1	0.9	0.9	

Hidden Markov Model



Given O and Q, we can compute:

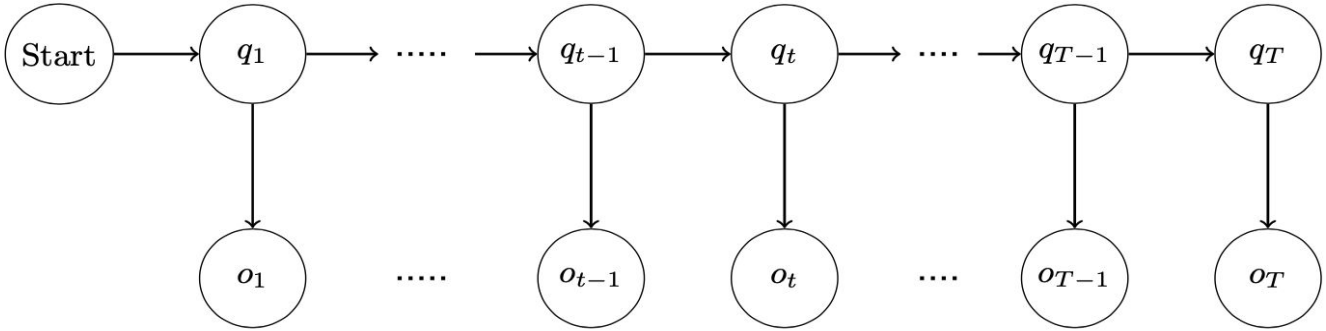
Pr(O|Q): product of $Pr(o_i \mid q_i)$

Pr(Q): product of $Pr(q_i \rightarrow q_{i+1})$

Pr(O, Q): Pr(O, Q)=Pr(O|Q)Pr(Q)

= product of all the $Pr(o_i \mid q_i)$ and $Pr(q_i \rightarrow q_{i+1})$

Hidden Markov Model

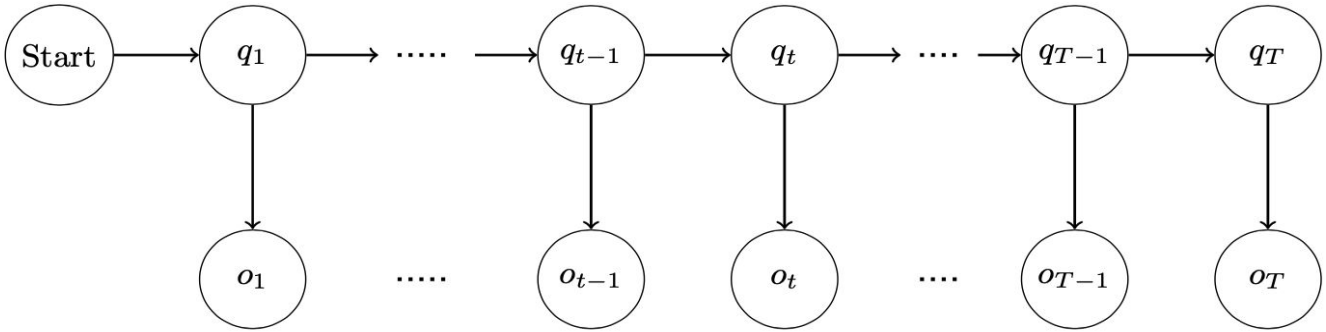


Very often, Q is unknown and we need to infer Q.
Detection/Matching/Decoding problem:

Given model parameters $\boldsymbol{\lambda}$ ($\boldsymbol{\lambda} = (\pi, A, B)$) and observation O, find the optimal Q which maximizes $\Pr(O,Q|\boldsymbol{\lambda})$.

Enumerate all the Q? Exponential size.
Viterbi algorithm

The Viterbi Algorithm to Find Best Path

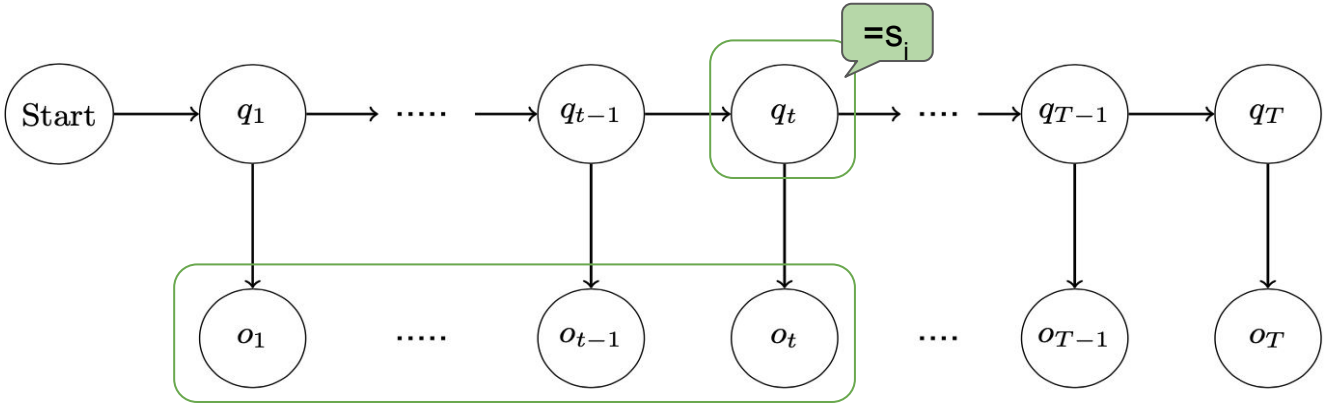


Dynamic programming to find the optimal Q which maximizes Pr(O,Q|λ)

Subproblem: the probability of the **best** path for $o_1 \ldots o_t$ that ends at state i.

$$w_t(i) = \max_{q_1,\ldots,q_{t-1}} Pr(o_1, o_2, \ldots, o_t, q_t = s_i)$$

The Viterbi Algorithm to Find Best Path



Dynamic programming to find the optimal Q which maximizes Pr(O,Q|λ)

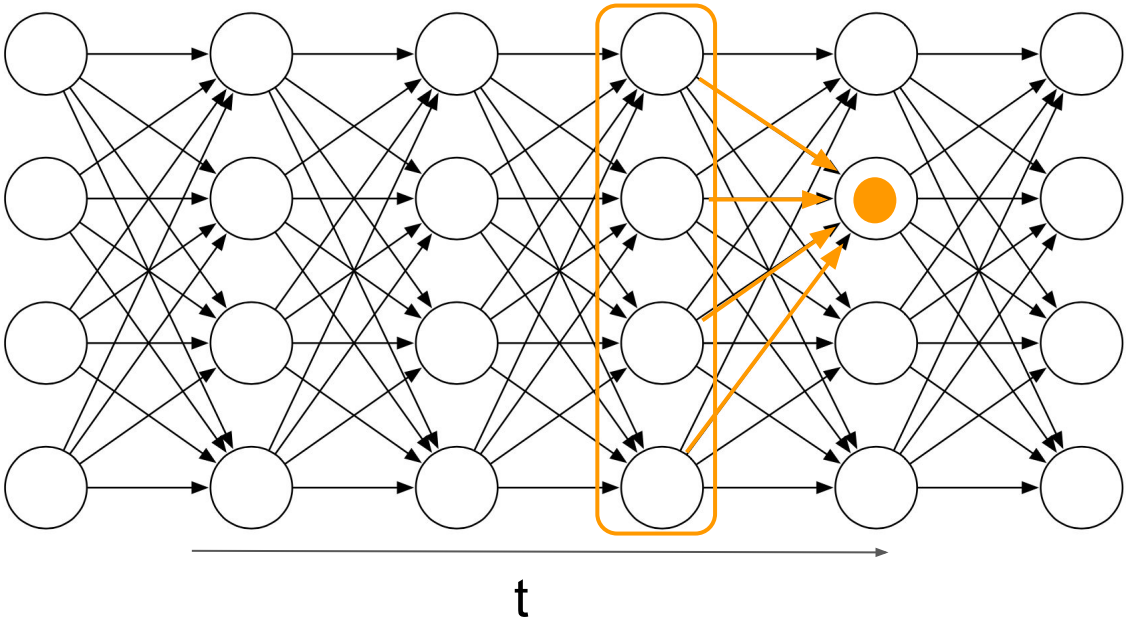
Subproblem: the probability of the **best** path for $o_1 \ldots o_t$ that ends at state i.

$$w_t(i) = \max_{q_1,\ldots,q_{t-1}} Pr(o_1, o_2, \ldots, o_t, q_t = s_i)$$

The Viterbi Algorithm to Find Best Path

$w_t(i)$: = the probability of the **best** path for $o_1 \ldots o_t$ that ends at state i.

Smaller subproblem: calculate $w_{t-1}(j)$

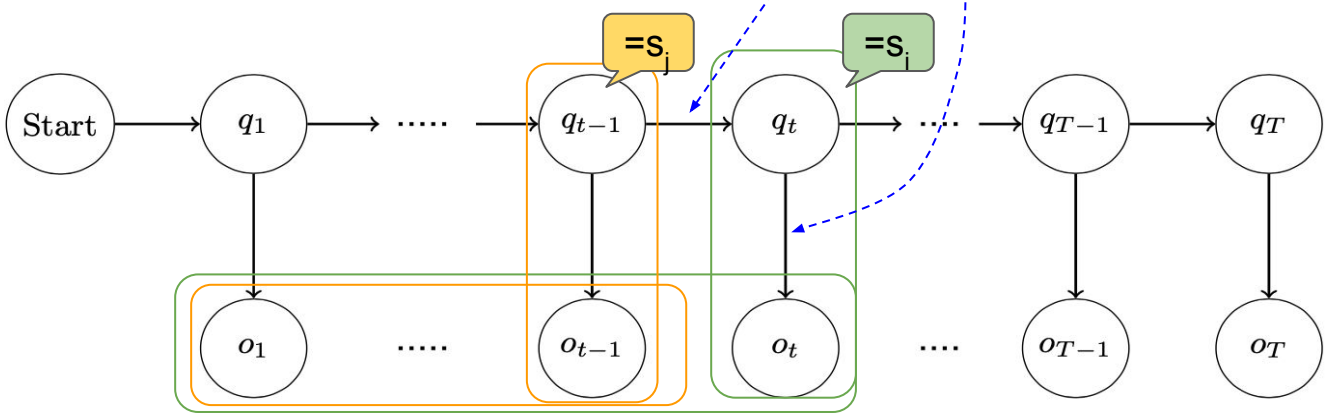


The Viterbi Algorithm to Find Best Path

$w_t(i)$: = the probability of the **best** path for $o_1 \ldots o_t$ that ends at state i.

Smaller subproblem: calculate $w_{t-1}(j)$

$$w_t(i) = \max_{1 \le j \le N} w_{t-1}(j) a_{ji} b_i(o_t)$$

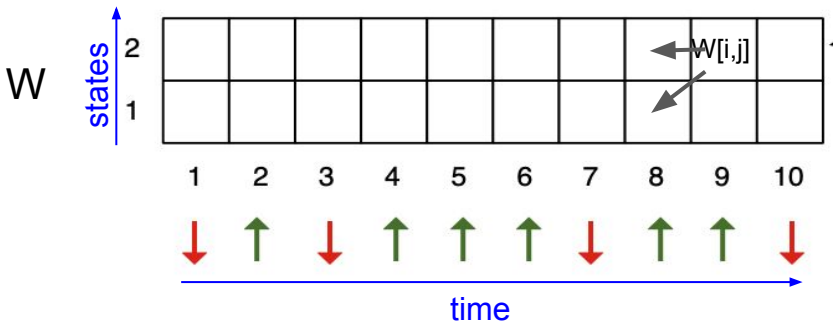


The Viterbi Algorithm to Find Best Path

$w_t(i)$: = the probability of the **best** path for $o_1 \ldots o_t$ that ends at state i.

Smaller subproblem: calculate $w_{t-1}(i)$

$$w_t(i) = \max_{1 \le j \le N} w_{t-1}(j) a_{ji} b_i(o_t)$$



The Viterbi Algorithm to Find Best Path

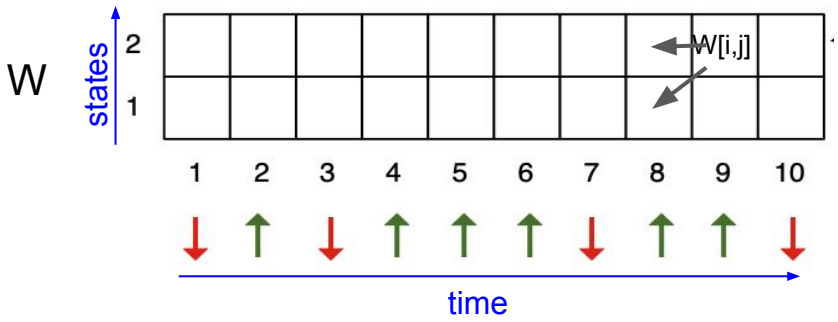
$w_t(i)$: = the probability of the **best** path for $o_1 \ldots o_t$ that ends at state i.

Recurrence:

$$w_t(i) = \max_{1 \le j \le N} w_{t-1}(j) a_{ji} b_i(o_t)$$

Base case:

$$w_1(i) = \pi_i b_i(o_1)$$



The Viterbi Algorithm to Find Best Path

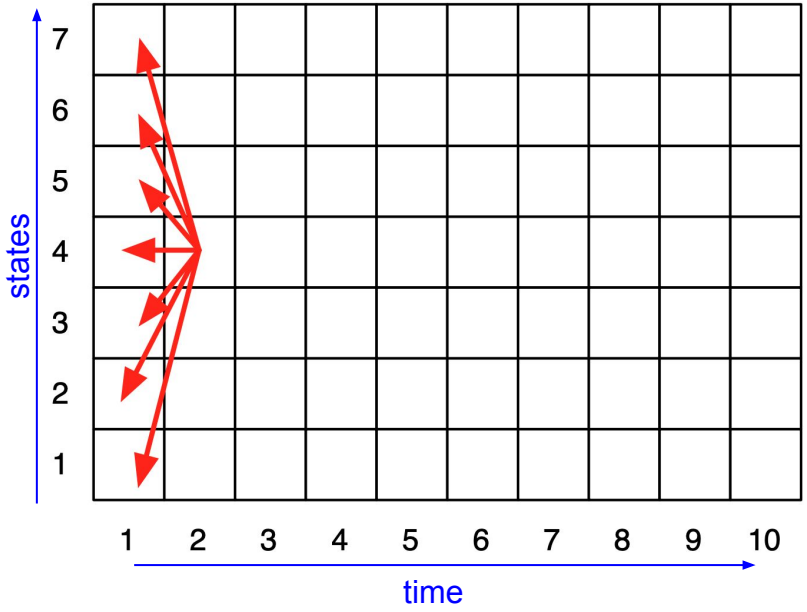
$w_t(i)$: = the probability of the **best** path for $o_1 \ldots o_t$ that ends at state i.

Recurrence:

$$w_t(i) = \max_{1 \le j \le N} w_{t-1}(j) a_{ji} b_i(o_t)$$

Base case:

$$w_1(i) = \pi_i b_i(o_1)$$



The Viterbi Algorithm to Find Best Path

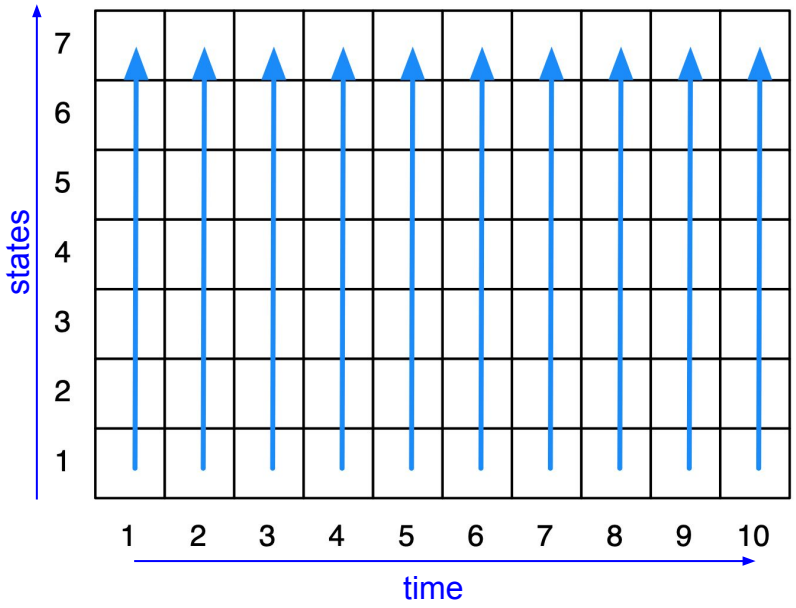
$w_t(i)$: = the probability of the **best** path for $o_1 \ldots o_t$ that ends at state i.

Recurrence:

$$w_t(i) = \max_{1 \le j \le N} w_{t-1}(j) a_{ji} b_i(o_t)$$

Base case:

$$w_1(i) = \pi_i b_i(o_1)$$



The Viterbi Algorithm to Find Best Path

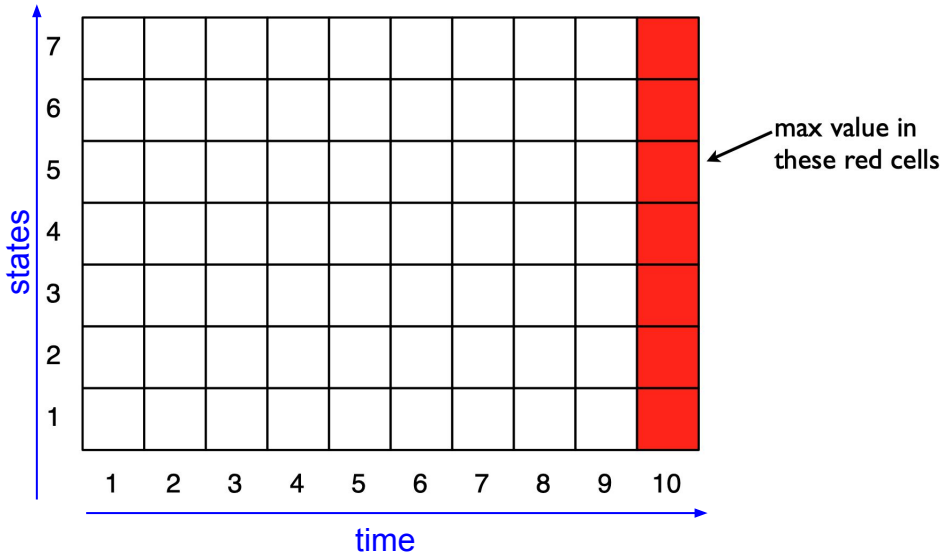
$w_t(i)$: = the probability of the **best** path for $o_1 \ldots o_t$ that ends at state i.

Recurrence:

$$w_t(i) = \max_{1 \le j \le N} w_{t-1}(j) a_{ji} b_i(o_t)$$

Base case:

$$w_1(i) = \pi_i b_i(o_1)$$



Running Time

- # of subproblems = $O(T|S|)$, where T is the length of the sequence.
- Time to solve a subproblem = $O(|S|)$
- Total running time: $O(T|S|^2)$

Using Logs

Typically, we take the log of the probabilities to avoid multiplying a lot of (small) terms:

$$\boxed{\log(ab) = \log(a) + \log(b)}$$

$$\begin{aligned}
\log(w_t(i)) &= \max_{1 \le j \le N} \{\log(w_{t-1}(j) \cdot a_{ji} \cdot b_i(o_t))\} \\
&= \max_{1 \le j \le N} \{\log(w_{t-1}(j)) + \log(a_{ji}) + \log(b_i(o_t))\}
\end{aligned}$$

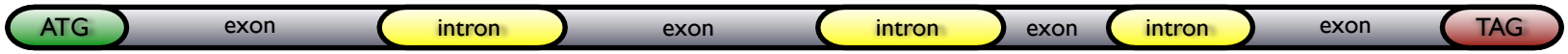
Why do we want to avoid multiplying lots of terms?

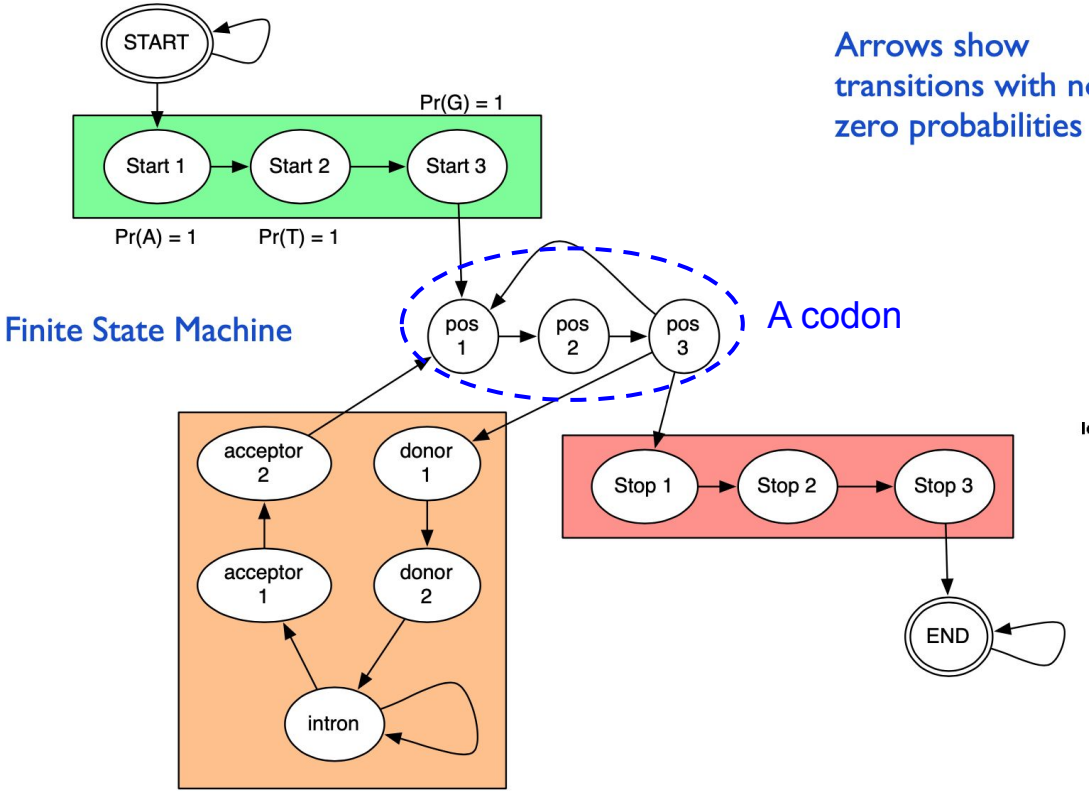
Multiplying leads to very small numbers: 0.1 x 0.1 x 0.1 x 0.1 x 0.1 = 0.00001

This can lead to underflow.

Taking logs and adding keeps numbers bigger.

An example of application in gene finding





Arrows show transitions with non-zero probabilities

Second Letter

1st letter	U		C		A		G		3rd letter
U	UUU	Phe	UCU	Ser	UAU	Tyr	UGU	Cys	U
	UUC		UCC		UAC		UGC		C
	UUA	Leu	UCA		UAA	Stop	UGA	Stop	A
	UUG		UCG		UAG	Stop	UGG	Trp	G
C	CUU	Leu	CCU	Pro	CAU	His	CGU	Arg	U
	CUC		CCC		CAC		CGC		C
	CUA		CCA		CAA	Gln	CGA		A
	CUG		CCG		CAG		CGG		G
A	AUU	Ile	ACU	Thr	AAU	Asn	AGU	Ser	U
	AUC		ACC		AAC		AGC		C
	AUA		ACA		AAA	Lys	AGA	Arg	A
	AUG	Met	ACG		AAG		AGG		G
G	GUU	Val	GCU	Ala	GAU	Asp	GGU	Gly	U
	GUC		GCC		GAC		GGC		C
	GUA		GCA		GAA	Glu	GGA		A
	GUG		GCG		GAG		GGG		G

The scoring problem with an HMM

Given an HMM (with known parameters **λ**), what's the probability of an observed sequence O being generated from this HMM?

$$Pr(O|\lambda) = \sum_{Q} Pr(O, Q|\lambda)$$

$$\lambda = (\pi, A, B)$$

$$\pi = \{\pi_i\} \qquad \pi_i = Pr(q_1 = s_i|\lambda), 1 \leq i \leq N \qquad \sum_{i=1}^{N} \pi_i = 1$$

$$A = \{a_{ij}\} \quad a_{ij} = Pr(q_{t+1} = s_j|q_t = s_i, \lambda), 1 \leq i, j \leq N \qquad \sum_{j=1}^{N} a_{ij} = 1, 1 \leq i \leq N$$

$$B = \{b_i(v_k)\} \quad b_i(v_k) = Pr(o_t = v_k|q_t = s_i, \lambda), 1 \leq i \leq N, 1 \leq k \leq M$$

$$\sum_{k=1}^{M} b_i(v_k) = 1, 1 \leq i \leq N$$

The scoring problem with an HMM

Given an HMM (with known parameters **λ**), what's the probability of an observed sequence O being generated from this HMM?

$$Pr(O|\lambda) = \sum_{Q} Pr(O, Q|\lambda)$$

Recall that in the decoding problem, we want to find Q*:

$$Q* = \text{argmax}_{Q} Pr(O, Q|\lambda)$$

The Forward algorithm

Recall

$$w_t(i) = \max_{q_1,\ldots,q_{t-1}} Pr(o_1, o_2, \ldots, o_t, q_t = s_i)$$

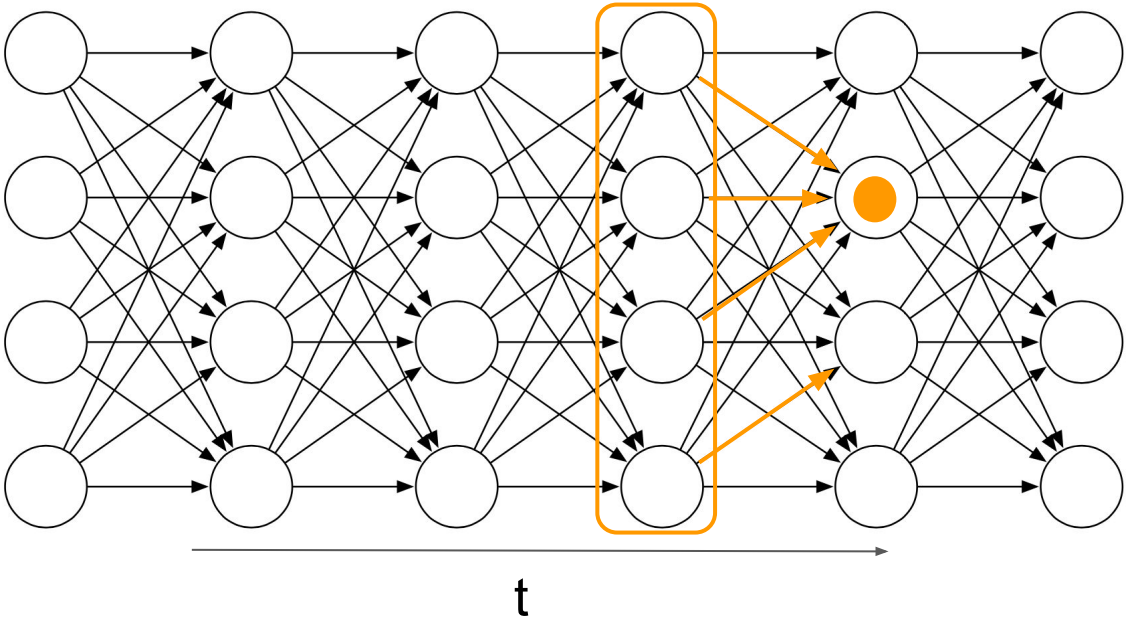
Now

$$\begin{aligned} \alpha_t(i) &= Pr(o_1, \ldots, o_t, q_t = s_i | \lambda) \\ &= \sum_{q_1,\ldots,q_{t-1}} Pr(o_1, \ldots, o_t, q_1, \ldots, q_{t-1}, q_t = s_i | \lambda) \end{aligned}$$

The Forward algorithm

	Base case	Recurrence
Viterbi	$w_1(i) = \pi_i b_i(o_1)$	$w_t(i) = \max_{1 \le j \le N} w_{t-1}(j) a_{ji} b_i(o_t)$
Forward	$\alpha_1(i) = \pi_i b_i(o_1)$	$\alpha_t(i) = \sum_{1 \le j \le N} \alpha_{t-1}(j) a_{ji} b_i(o_t)$

The Forward algorithm



$$\alpha_t(i) = \sum_{1 \leq j \leq N} \alpha_{t-1}(j) a_{ji} b_i(o_t)$$

The training problem

When $\boldsymbol{\lambda}$ is unknown, we need to learn $\boldsymbol{\lambda}$ from data.

- When both observation sequence O and state sequence Q are known
  Maximum likelihood estimation

- When O is known but Q is unknown
  Expectation-Maximization (EM) algorithm

Further readings

- Ghahramani, Z. Probabilistic machine learning and artificial intelligence. *Nature* **521**, 452–459 (2015)

- R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*,
  pp. 46–66 for algorithms on a standard HMM model including parameter estimation;
  pp. 102–113 for constructing a profile HMM and Forward and Viterbi algorithms on a profile HMM;
  pp. 149–154 for applying profile HMM to MSA, and the training (parameter estimation) of a profile HMM.
  pp. 323–325 for the EM (expectation maximization) algorithm.