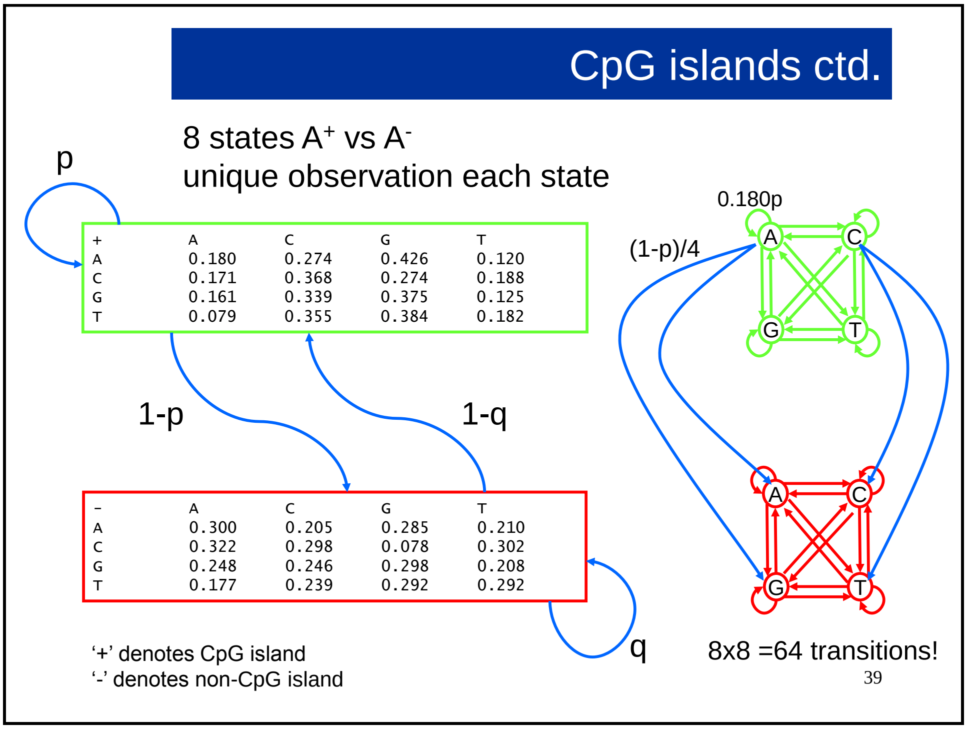


CpG islands ctd.

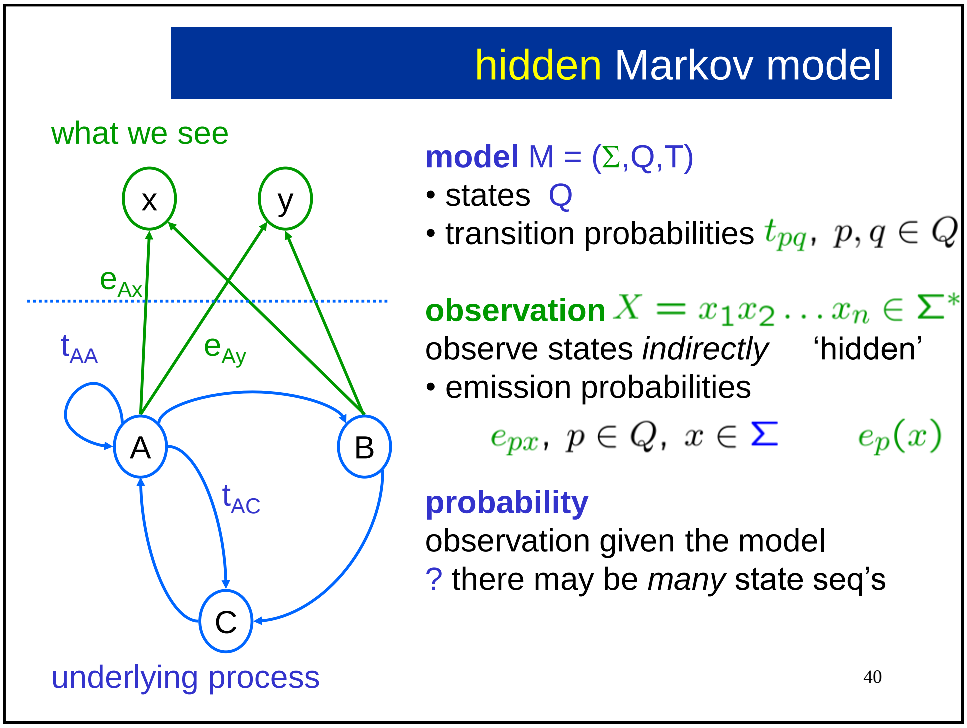
8 states A^+ vs A^-
unique observation each state

+	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

-	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

‘+’ denotes CpG island
‘-’ denotes non-CpG island

8x8 =64 transitions!



hidden Markov model

model M = (Σ,Q,T)
- states Q
- transition probabilities t_{pq}, $p, q \in Q$

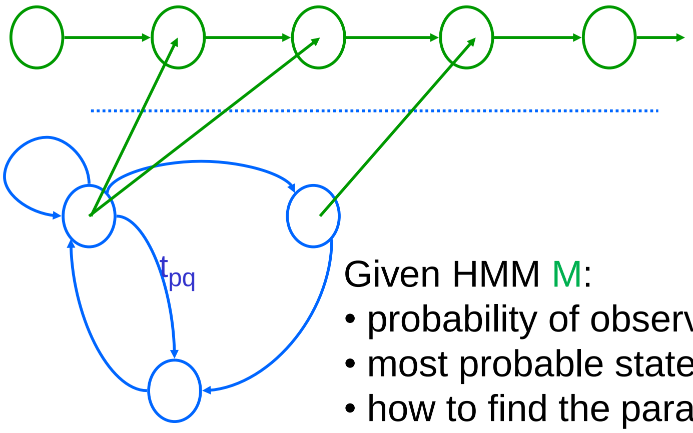
observation $X = x_1 x_2 \ldots x_n \in \Sigma^*$
observe states *indirectly* ‘hidden’
- emission probabilities

$$e_{px},\ p \in Q,\ x \in \Sigma \qquad e_p(x)$$

probability
observation given the model
? there may be *many* state seq’s

HMM main questions

observation $X \in \Sigma^*$

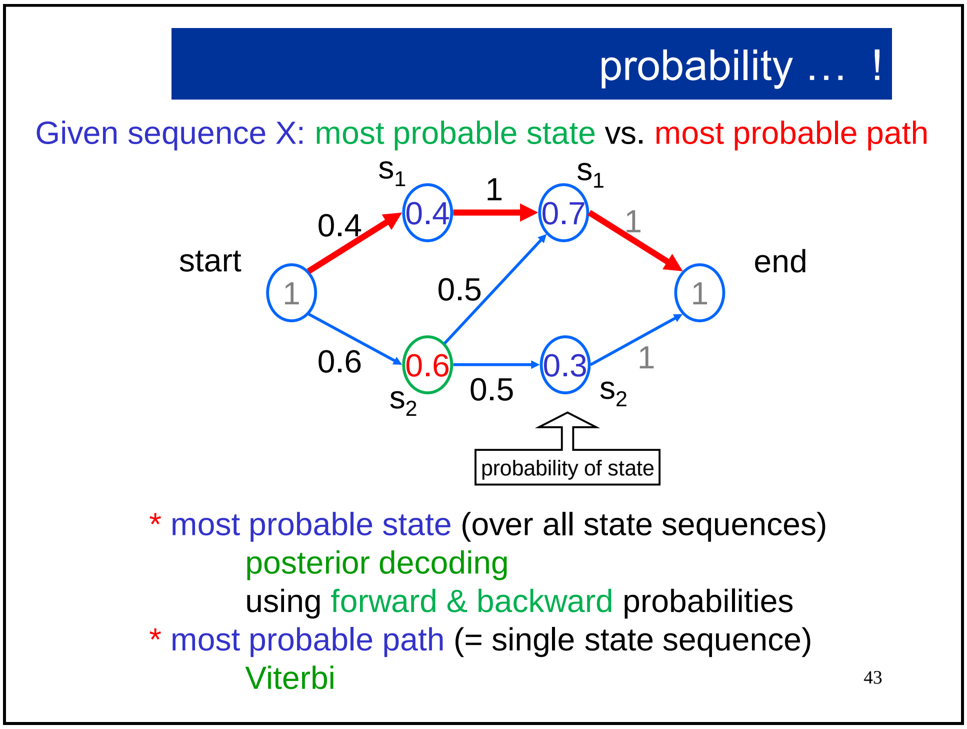


Given HMM M:

- probability of observation X?
- most probable state sequence?
- how to find the parameters of the model M? *training*

Three Important Questions
(See also L.R. Rabiner (1989))

- How likely is a given sequence?
  - The Forward algorithm (probability over all paths)
- What is the most probable “path” for generating a given sequence?
  - The Viterbi algorithm
- How can we learn the HMM parameters given a set of sequences?
  - The Forward-Backward (Baum-Welch) algorithm

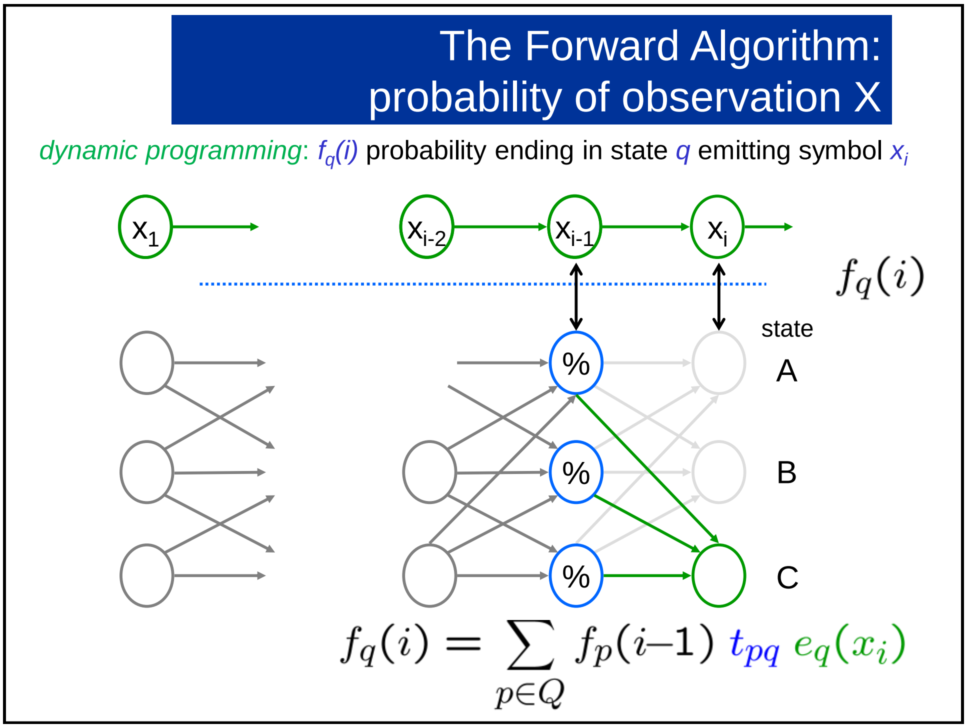


probability … !

Given sequence X: most probable state vs. most probable path

* most probable state (over all state sequences)
posterior decoding
using forward & backward probabilities

* most probable path (= single state sequence)
Viterbi



The Forward Algorithm: probability of observation X

dynamic programming: $f_q(i)$ probability ending in state q emitting symbol x_i

$$f_q(i) = \sum_{p \in Q} f_p(i{-}1)\ t_{pq}\ e_q(x_i)$$

The Forward Algorithm: probability of observation X

probability observing $x_1, \ldots, x_i$ and ending in state q:

$$f_q(i) = P(x_1 \ldots x_i, \pi_i = q)$$

$$f_q(i) = \sum_{p \in Q} f_p(i-1)\; t_{pq}\; e_q(x_i)$$

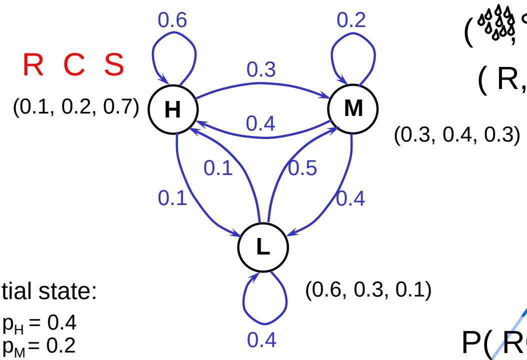
'forward' probability

$$P(X) = \sum_{p \in Q} f_p(n)\; t_{p*}$$

* = end-state

45

Probability of observation: weather



(🌧, ☁, ☀)

(R, C, S)

Transitions:

- Remain in H
- Coming from M
- Coming from L

Initial state:

$p_H = 0.4$
$p_M = 0.2$
$p_L = 0.4$

P(R...)

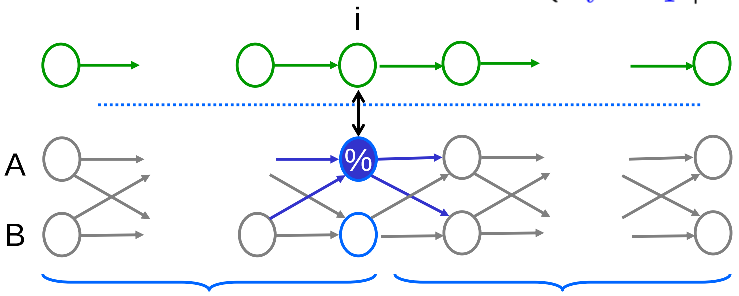
P(RCCSS) = P(RC…)

			1:R	2:C
	H	0	4·1 = 4	(4·6 +6·4 +24·1)·2 = 144 (x10^{-4})
	M	0	2·3 = 6	(4·3 +6·2 +24·5)·4 = 576 (x10^{-4})
	L	0	4·6 = 24	(4·1 +6·4 +24·4)·3 = 372 (x10^{-4})
Start:	0	1		

46

HMM: posterior decoding

Given X the prob. that the *i-th* state equals *q:* $P(\pi_i = q \mid X)$



$f_q(i) = P(x_1 \ldots x_i, \pi_i = q)$ $\quad$ $b_q(i) = P(x_{i+1} \ldots x_n \mid \pi_i = q)$

$\xrightarrow{\text{forward}}$ $\quad$ $\xleftarrow{\text{backward}}$

$P(X, \pi_i = q) = f_q(i) b_q(i)$ => $\boxed{P(\pi_i = q \mid X) = \frac{f_q(i) b_q(i)}{P(X)}}$

Posterior Decoding Problem

Posterior Decoding is another decoding method:

Input:

Given a Hidden Markov Model M = (Σ, Q, Θ) and a sequence X for which the generating path P is unknown.

Question:

For each 1 ≤ i ≤ L (the length of the path P) and state q in Q compute the probability: $P(\pi_i = q \mid X)$.

Posterior Decoding Problem

$P(\pi_i = q \mid X)$ gives two additional decoding possibilities:

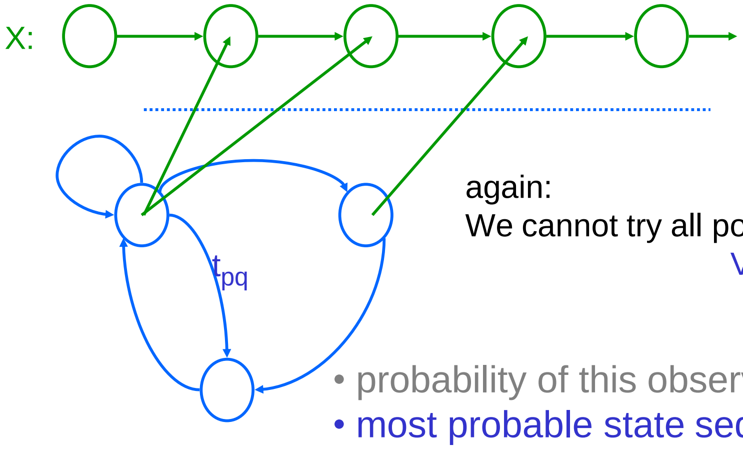
1. Alternative ‘path’ P* that follows the max probability states: $\mathrm{argmax}_{\text{state } q} \{ P(\pi_i = q \mid X) \}$.
2. Define a function $g(q)$ on the states q in Q, then $G(i \mid X) = \sum_q \{ P(\pi_i = q \mid X) \cdot g(q) \}$

We can use 2) to calculate the posterior probability of each nucleotide of X to be in a CpG-island, using a function g(q) defined on all states q in Q:

$$g(q) = \begin{cases} 1 \text{ for all } q \text{ that are CpG-island states,} \\ 0 \text{ otherwise.} \end{cases}$$

HMM main questions

observation $X \in \Sigma^*$ ⇒ most probable state sequence



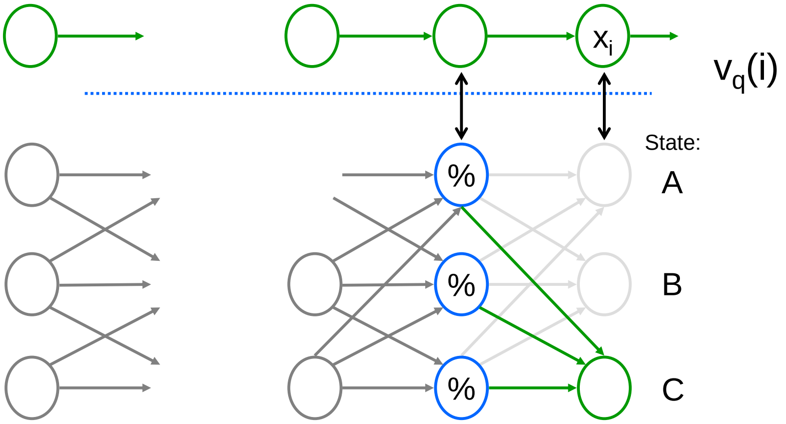
again:
We cannot try all possibilities
Viterbi

- probability of this observation?
- most probable state sequence?
- how to find the model? *training*

Viterbi algorithm

most probable state sequence for observation *X*

(1) *dynamic programming*: $v_q(i)$ probability ending in state q and emitting x_i



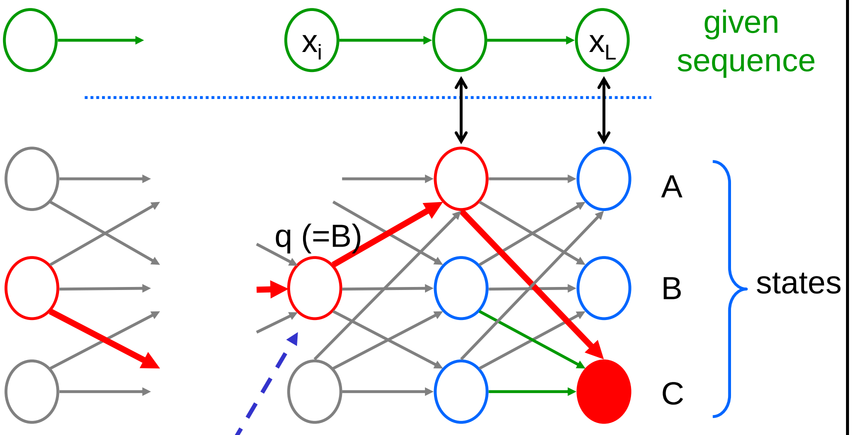
$$v_q(i) = \max_{p\in Q} v_p(i-1)\; t_{pq}\; e_q(x_i)$$

51

Decoding Problem: The Viterbi algorithm

(1) *dynamic programming*: max probability ending in state

(2) *traceback*: most probable state sequence



$$\boxed{v_q(i) = \max_{p\in Q} v_p(i-1)\; t_{pq}\; e_q(x_i)}$$

52

HMM Decoding: two explanations

posterior Σ

best state every position

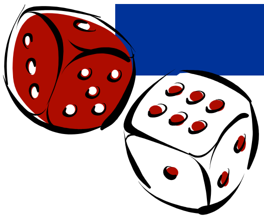
But: path may not be allowed by model

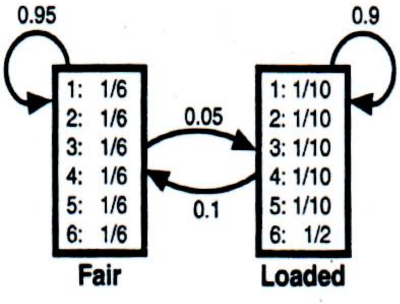
viterbi max

optimal global path

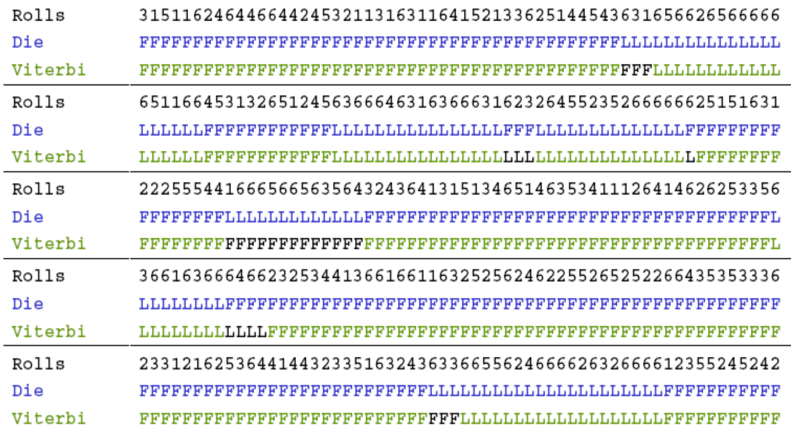
But: many paths with similar probability

dishonest casino dealer





dishonest casino dealer



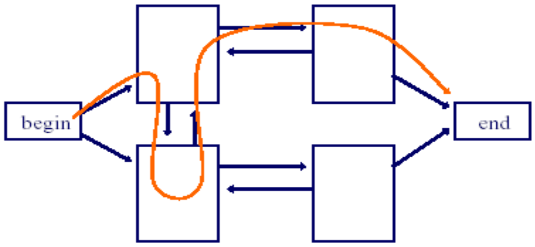
```
Rolls    315116246446644245321131631164152133625144543631656626566666
Die      FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLL
Viterbi  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLL
Rolls    651166453132651245636664631636663162326455236266666625151631
Die      LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLFFFLLLLLLLLLLLLLLFFFFFFFFF
Viterbi  LLLLLLFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLFFFFFFFF
Rolls    222555441666566563564324364131513465146353411126414626253356
Die      FFFFFFFFLLLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL
Viterbi  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFL
Rolls    366163666466232534413661661163252562462255265252266435353336
Die      LLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Viterbi  LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
Rolls    233121625364414432335163243633665562466662632666612355245242
Die      FFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
Viterbi  FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFLLLLLLLLLLLLLLLLLLLFFFFFFFFFFF
```

dishonest casino dealer

```
   Observation
366163666466232534413661661163252562462255265252266435353336
   Viterbi
LLLLLLLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Compare to:

   Forward
FFLLLLLLLLLLLLFFFFFFFFLFLLLFLLFFFFFFFFFFFFFFFFFFFFLFFFFFFFFF

   Posterior (total)
LLLLLLLLLLLLFFFFFFFFFLLLLLLFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
```

Learning if correct path is known

- Learning is simple if we **know the correct path** for each sequence in our training set



- estimate parameters by counting the number of times each parameter is used across the training set

Sketch: Parameter estimation

training sequences $X^{(i)}$
optimize score $\prod_{i=1}^{n} P(X^{(i)} \mid \Theta)$ for model Θ.

If state sequences are known

- count transitions *pq* A_{pq}
- count emissions *b* in *p* $E_p(b)$

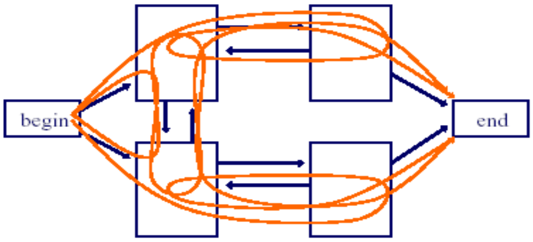
divide by

- total transitions in *p*
- emissions in *q*

Laplace correction for dealing with ‘zero’ probabilities.
Adding 1 to each count.

Learning With Hidden State

- If we **don’t know the correct path** for each sequence in our training set, consider all possible paths for the sequence



- Estimate parameters through a procedure that counts the expected number of times each parameter is used across the training set.

Learning Parameters: The Baum-Welch Algorithm

- Here we use the Forward-Backward algorithm
- An Expectation Maximization (EM) algorithm
  - EM is a family of algorithms for learning probabilistic models in problems that involve hidden states
- In this context, the hidden state is the path that best explains each training sequence.

- Note, finding the parameters of the HMM that optimally explains the given sequences is NP-Complete!

HMM: state sequences unknown: Baum-Welch

Baum-Welch training

- Based on given HMM Θ
- Given a training set of sequences X

- Determine:
  - expected number of transitions and
  - expected number of emissions
- Apply ML and build a new (better) model:
  - ML tries to find a model that gives the training data the highest likelihood
- Iterate until convergence.

Note:

- can get stuck in local maxima
- does not understand the semantics of the states

Baum-Welch Re-estimation

For the re-estimation we need the expected counts
For the transitions and the emissions in the HMM:

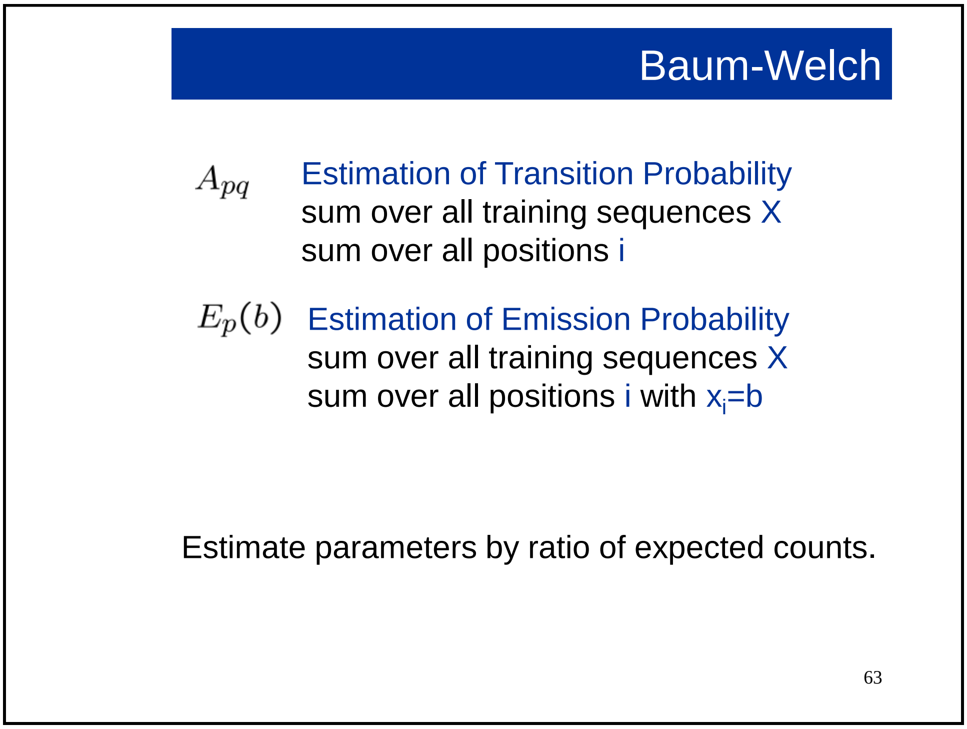
- Apply the backward-forward algorithm.

Probability of state q when emitting X_i:

$$P(\pi_i = q \mid X) = \frac{f_q(i) b_q(i)}{P(x)}$$

Probability of transition (p,q) after emitting X_i:

$$P(\pi_i = p, \pi_{i+1} = q \mid X, \Theta) = \frac{f_p(i) \cdot t_{pq} \cdot e_q(x_{i+1}) \cdot b_q(i+1)}{P(X)}$$

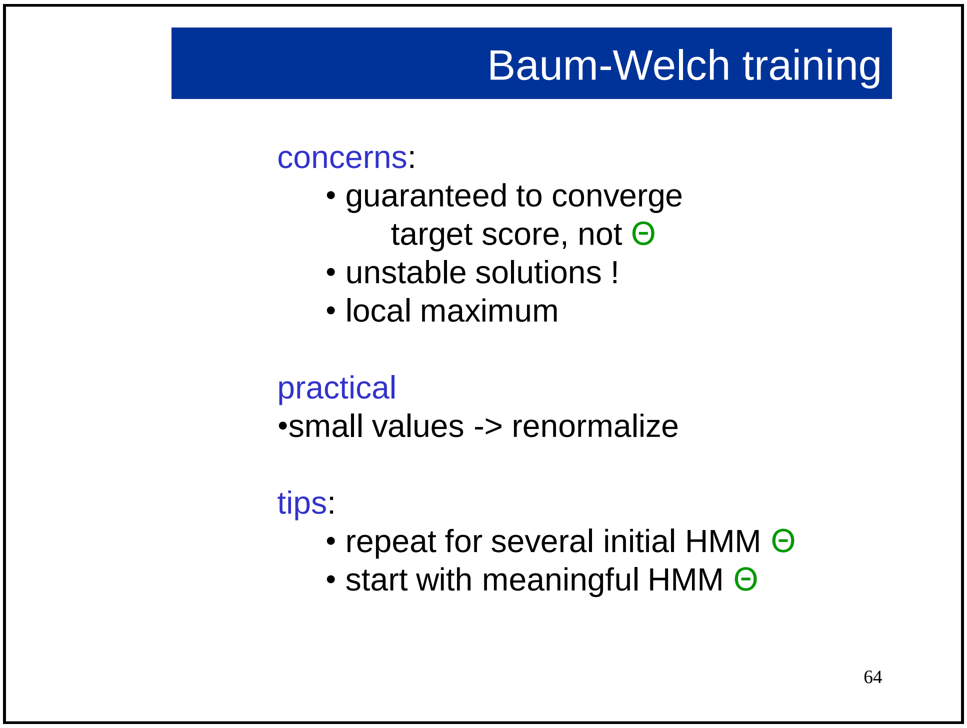


Baum-Welch

A_{pq} Estimation of Transition Probability
sum over all training sequences X
sum over all positions i

$E_p(b)$ Estimation of Emission Probability
sum over all training sequences X
sum over all positions i with x_i=b

Estimate parameters by ratio of expected counts.



Baum-Welch training

concerns:

- guaranteed to converge
  target score, not Θ
- unstable solutions !
- local maximum

practical
•small values -> renormalize

tips:

- repeat for several initial HMM Θ
- start with meaningful HMM Θ

Viterbi training

Viterbi training (sketch):

- determine optimal paths
- use these paths to re-compute parameters as in the case where paths are known
- score may decrease!

Computational Complexity of HMM Algorithms

- Given an HMM with S states and a sequence of length L, the complexity of the Forward, Backward and Viterbi algorithms is

$$O(S^2L)$$

  - This assumes that the states are densely interconnected
- Given M training sequences of length L, the complexity of Baum Welch on each iteration is

$$O(MS^2L)$$

Important Papers on HMM

L.R. Rabiner, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition,

Proceeding of the IEEE, Vol. 77, No. 22, February 1989.

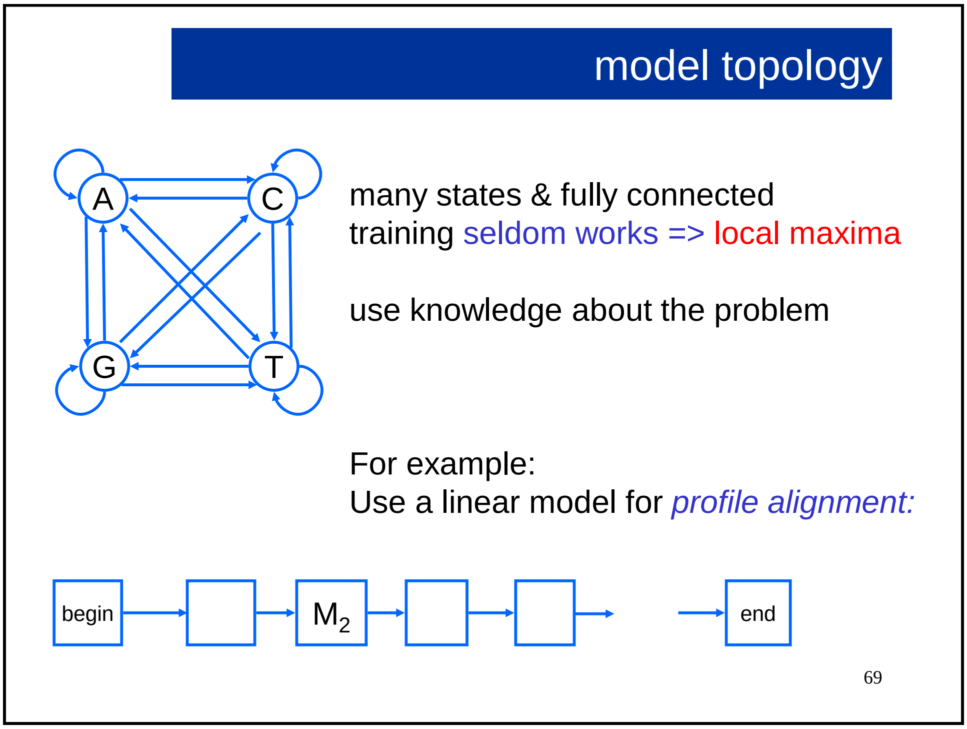
Krogh, I. Saira Mian, D. Haussler, A Hidden Markov Model that finds genes in E. coli DNA, Nucleid Acids Research, Vol. 22 (1994), pp 4768-4778

Furthermore:

R. Hassan, A combination of hidden Markov model and fuzzy model for stock market forecasting, Neurocomputing archive, Vol. 72 , Issue 16-18, pp 3439-3446, October 2009.

Hidden Markov Models

Applications

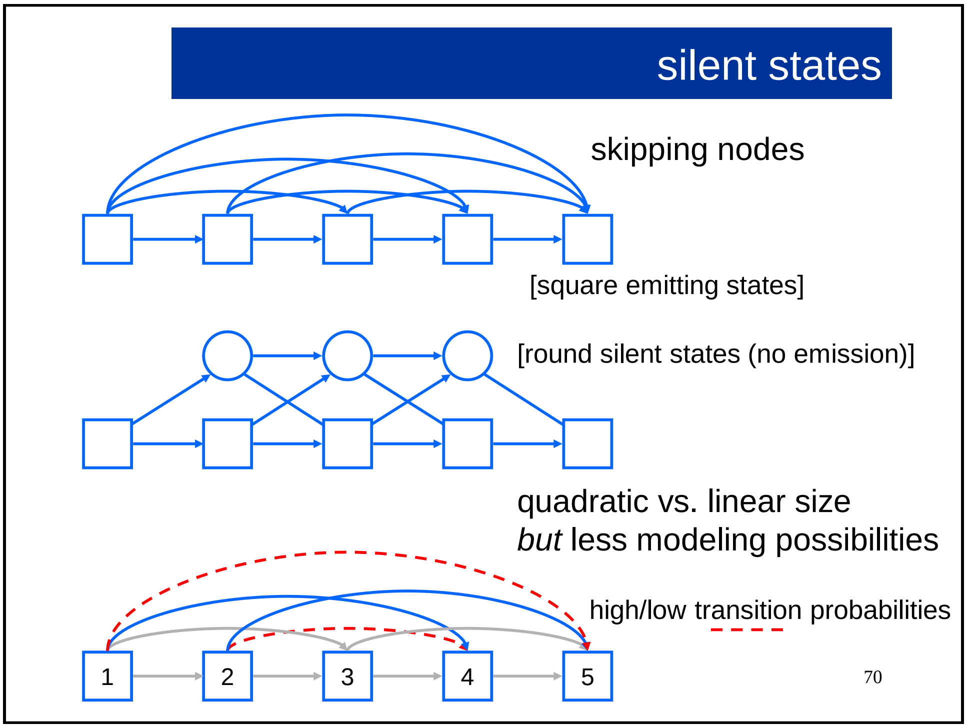


model topology

many states & fully connected
training seldom works => local maxima

use knowledge about the problem

For example:
Use a linear model for *profile alignment:*



silent states

skipping nodes

[square emitting states]

[round silent states (no emission)]

quadratic vs. linear size
but less modeling possibilities

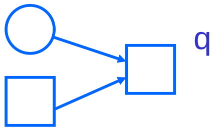
high/low transition probabilities

silent states: algorithm

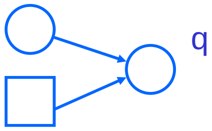
Previously: forward algorithm $$f_q(i) = \sum_{p \in Q} f_p(i-1)\, t_{pq}\, e_q(x_i)$$

From state p to state q

transition / emission



For emitting states q
=> calculated as before

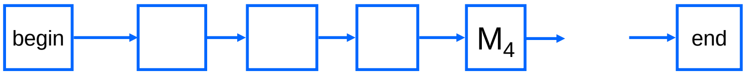


But for silent states q

$$f_q(i) = \sum_{p \in Q} f_p(i)\, t_{pq}$$

- no silent loops (!):
- update in ‘topological order’

profile alignment (no gaps)



Assume a given profile set:

```
12345678
VGAHAGEY
VTGNVDEV
VEADVAGH
VKSNDVAD
VYSTVETS
FNANIPKH
IAGNGAGV
```

=> Emission probability distribution function at state 4

No gaps
transition probabilities: 1
trivial alignment HMM to sequence

profile HMM $\mathcal{P}$ ‘dedicated topology’

$$X = x_1 x_2 \ldots x_L$$

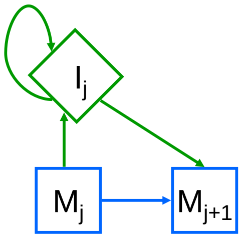
Let $e_i(b)$ be equal to the probability of observing symbol b at pos i, then:

$$P(X|\mathcal{P}) = \prod_{i=1}^{L} e_i(x_i)$$

profile alignment with gaps

Given profile sequences:

```
VGA--HAGEY
VNA--NVDEV
VEA--DVAGH
VKG--NYDED
VYS--TYETS
FNA--NIPKH
IAGADNGAGV
123__45678
```



insert state

match states

Emission probability distribution based on:
- background probabilities: $e_i(b) = p(b)$
- or based on alignment (match)

affine model

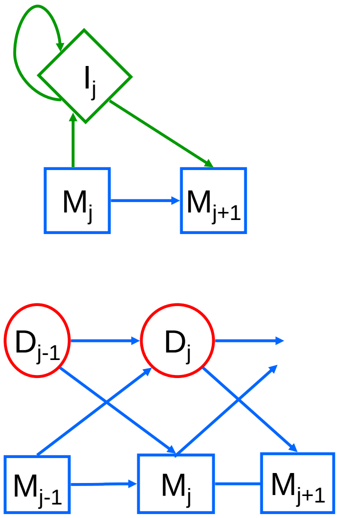
$$t_{M_jI_j} \cdot t_{I_jM_{j+1}} \cdot t_{I_jI_j}^{h-1}$$

open gap extension

profile alignment with gaps and deletes

Given profile Sequences:

```
VGA--HAGEY
V----NVDEV
VEA--DVAGH
VKG-----D
VYS--TYETS
FNA--NIPKH
IAGADNGAGV
123__45678
```



insert state

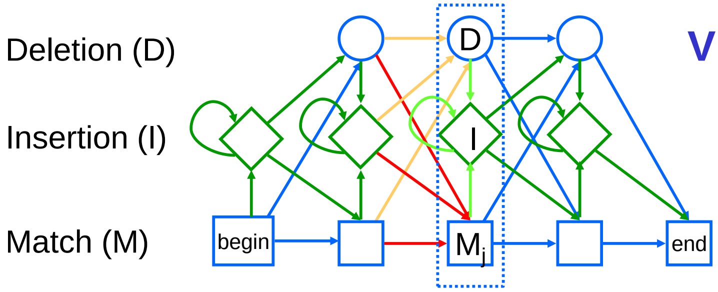
match states

delete state
(*silent*)

adapt Viterbi =>

HMM for profiles / multiple alignment

Viterbi


Deletion (D)

Insertion (I)

Match (M)

$$v_j^M(i) = e_{M_j}(x_i).\max_{Y=M,I,D} v_{j-1}^Y(i-1).\boxed{t_{Y_{j-1}M_j}}$$

$$v_j^I(i) = p(x_i).\max_{Y=M,I,D} v_j^Y(i-1).\boxed{t_{Y_{j-1}I_j}}$$

same level

$$v_j^D(i) = \max_{Y=M,I,D} v_{j-1}^Y(i).\boxed{t_{Y_{j-1}D_j}}$$

same position

profile alignment

Given a multiple alignment
with Insertion / Deletion states

```
VGA--HAGEY
V----NVDEV
VEA--DVAGH
VKG------D
VYS--TYETS
FNA--NIPKH
IAGADNGAGV
123  45678
```

Example counting for state 1:

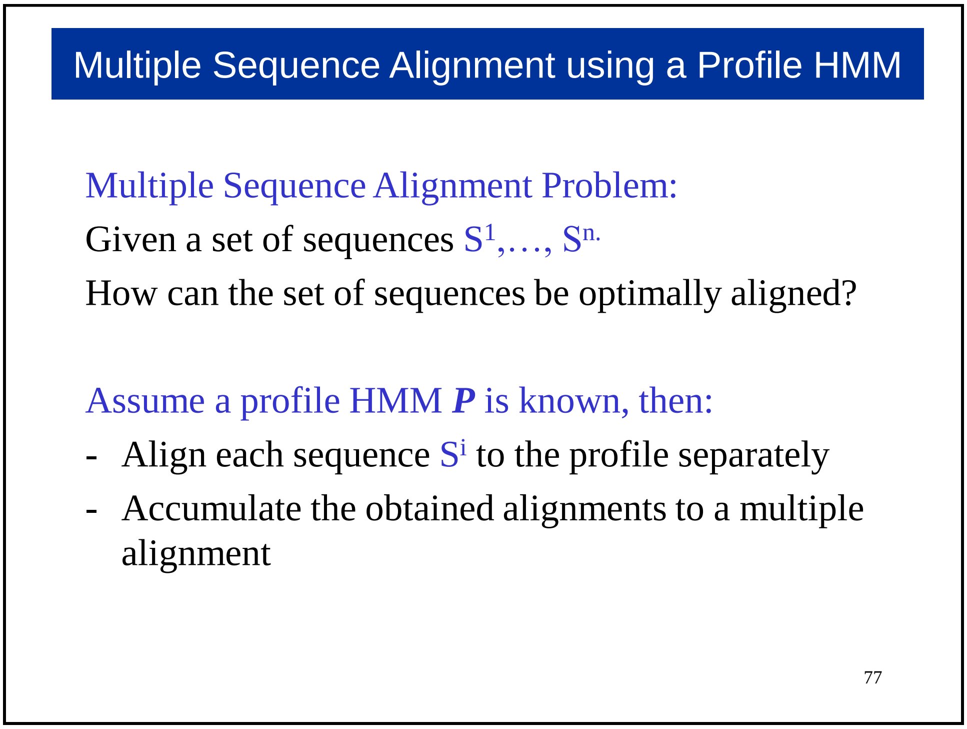
transitions

$M_1 \rightarrow M_2$	6+1	$7/_{10}$
$M_1 \rightarrow I_1$	0+1	$1/_{10}$
$M_1 \rightarrow D_1$	1+1	$2/_{10}$

Emissions **(for 20 Amino Acids)**

F	1+1	$2/_{27}$
I	1+1	$2/_{27}$
V	5+1	$6/_{27}$
other 17x	0+1	$1/_{27}$

Laplace correction, i.e., adding 1 for each frequency to avoid dividing by 0



Multiple Sequence Alignment using a Profile HMM

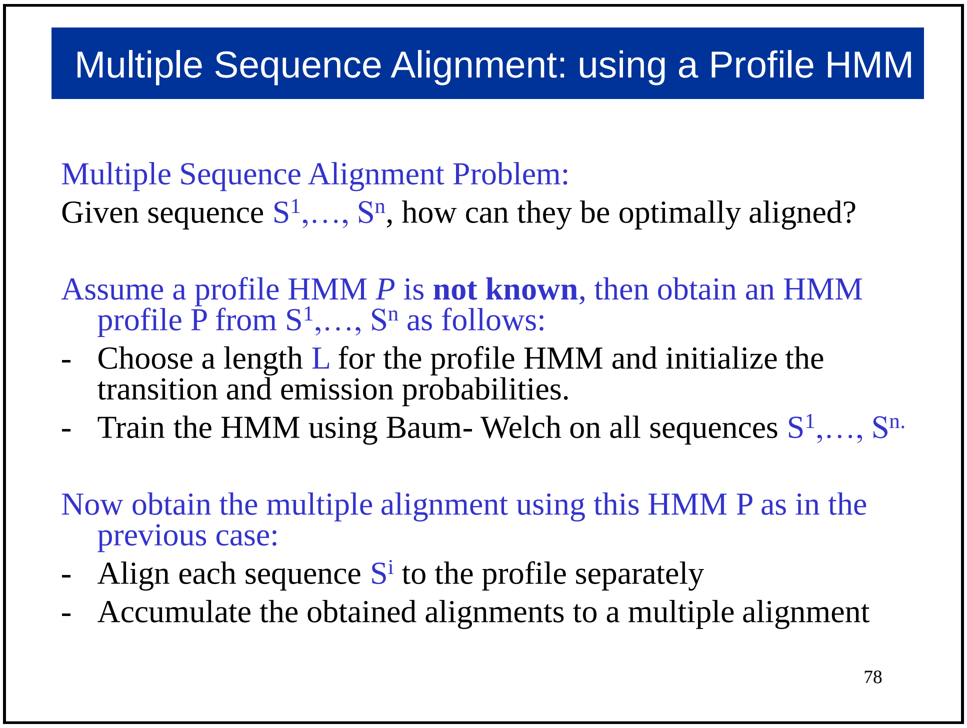
Multiple Sequence Alignment Problem:

Given a set of sequences S^1,…, S^n.

How can the set of sequences be optimally aligned?

Assume a profile HMM ***P*** is known, then:

- Align each sequence S^i to the profile separately
- Accumulate the obtained alignments to a multiple alignment



Multiple Sequence Alignment: using a Profile HMM

Multiple Sequence Alignment Problem:

Given sequence S^1,…, S^n, how can they be optimally aligned?

Assume a profile HMM *P* is **not known**, then obtain an HMM profile P from S^1,…, S^n as follows:

- Choose a length L for the profile HMM and initialize the transition and emission probabilities.
- Train the HMM using Baum- Welch on all sequences S^1,…, S^n.

Now obtain the multiple alignment using this HMM P as in the previous case:

- Align each sequence S^i to the profile separately
- Accumulate the obtained alignments to a multiple alignment

multiple alignment with profile

```
IAGADNGAGV
123II45678

VGAHAGEY
12345678

FNAPNI-KH
123I45678
     D
```

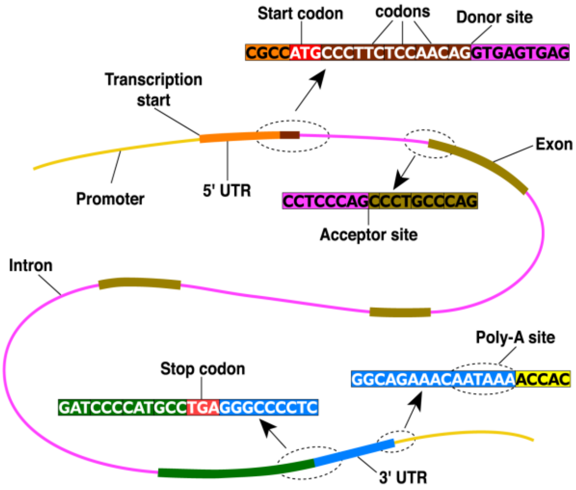
align each sequence separately

accumulate alignments M and D positions

```
VGA--HAGEY
FNAP-NI-KH
IAGADNGAGV
123  45678
```

align inserts (I) leftmost i positions

Gene Finding


Start codon
codons
Donor site
CGCCATGCCCTTCTCCAACAGGTGAGTGAG
Transcription start
Exon
Promoter
5' UTR
CCTCCCAGCCCTGCCCAG
Acceptor site
Intron
Poly-A site
Stop codon
GGCAGAAACAATAAAACCAC
GATCCCCATGCCTGAGGGCCCCTC
3' UTR