PROBABILISTIC MODELS: HIDDEN MARKOV MODELS

PROLOGUE:

Pitfalls of standard alignments

Scoring a pairwise alignment

A: ALA**E**VLIRLIT**K**LYP

B: ASA**K**HLNRLIT**E**LYP

$$Score(A,B) = \sum s(A^i, B^i)$$

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

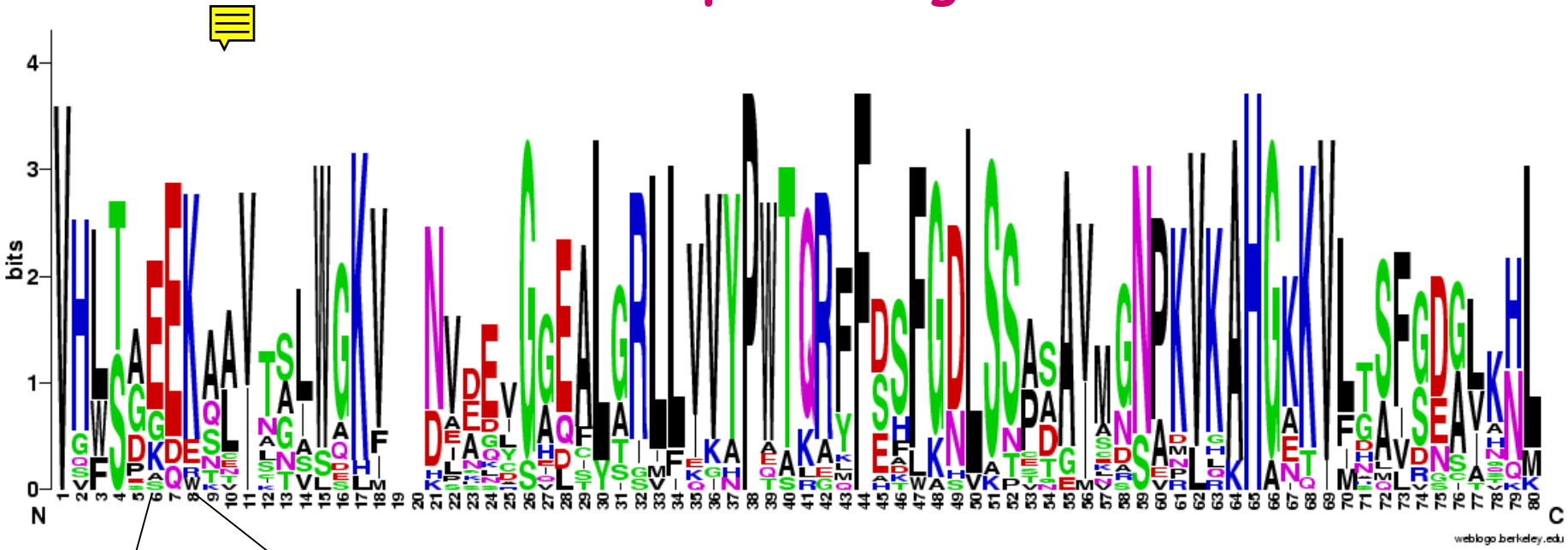
Blosum62

Alignment of a family (globins)

```
                                 10        20        30        40        50        60        70        80
lqb1 pea/1-471   -GFTDKQEALVNSSSE-FKQNLPGYSILFYTIVLEKAPAAKGLFSFLKD---TAGVEDSPKLQAHAEQVFGLVRDSAAQL
lqb1 vicfa/1-471-GFTEKQEALVNSSSQLFKQNPSNYSVLFYTIILQKAPTAKAMFSFLKD---SAGVVDSPKLGAHAEKVFGMVRDSAVQL
hbb speci/1-471 VHLSDGEKNAISTAWGKV--HAAEVGAEALGRLLVVYPWTQRFFDSFGDLSSASAVMGNAKVKAHGKKVIDSFSNGLKHL
hbb speto/1-471 VHLTDGEKNAISTAWGKV--NAAEIGAEALGRLLVVYPWTQRFFDSFGDLSSASAVMGNAKVKAHGKKVIDSFSNGLKHL
hbb equhe/1-471 VQLSGEEKAAVLALWDKV--NEEEVGGEALGRLLVVYPWTQRFFDSFGDLSNPAAVMGNPKVKAHGKKVLHSFGEGVHHL
hbb sunmu/1-471 VHLSGEEKACVTGLWGKV--NEDEVGAEALGRLLVVYPWTQRFFDSFGDLSSASAVMGNPKVKAHGKKVLHSLGEGVANL
hbb tupql/1-471 VHLSGEEKAAVTGLWGKV--DLEKVGGQSLGSLLIVYPWTQRFFDSFGDLSSPSAVMSNPKVKAHGKKVLTSFSDGLNHL
hbb calar/1-471 VHLTGEEKSAVTALWGKV--NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMNNPKVKAHGKKVLGAFSDGLTHL
hbb mansp/1-471 VHLTPEEKTAVTTLWGKV--NVDEVGGEALGRLLVVYPWTQRFFDSFGDLSSPDAVMGNPKVKAHGKKVLGAFSDGLNHL
hbb rabit/1-471 VHLSSEEKSAVTALWGKV--NVEEVGGEALGRLLVVYPWTQRFFESFGDLSSANAVMNNPKVKAHGKKVLAAFSEGLSHL
hbb ursma/1-471 VHLTGEEKSLVTGLWGKV--NVDEVGGEALGRLLVVYPWTQRFFDSFGDLSSADAIMNNPKVKAHGKKVLNSFSDGLKNL
hbb triin/1-471 VHLTPEEKALVIGLWAKV--NVKEYGGEALGRLLVVYPWTQRFFEHFGDLSSASAIMNNPKVKAHGEKVFTSFGDGLKHL
hbb ornan/1-471 VHLSGGEKSAVTNLWGKV--NINELGGEALGRLLVVYPWTQRFFEAFGDLSSAGAVMGNPKVKAHGAKVLTSFGDALKNL
hbb tacac/1-471 VHLSGSEKTAVTNLWGHV--NVNELGGEALGRLLVVYPWTQRFFESFGDLSSADAVMGNAKVKAHGAKVLTSFGDALKNL
hbe ponpy/1-471 VHFTAEEKAAVTSLWSKM--NVEEAGGEALGRLLVVYPWTQRFFDSFGNLSSPSAILGNPKVKAHGKKVLTSFGDAIKNM
hbb colli/1-471 VHWSAEEKQLITSIWGKV--NVADCGAEALARLLIVYPWTQRFFSSFGNLSSATAISGNPNVKAHGKKVLTSFGDAVKNL
hbb larri/1-471 VHWSAEEKQLITGLWGKV--NVADCGAEALARLLIVYPWTQRFFASFGNLSSPTAINGNPMVRAHGKKVLTSFGEAVKNL
hbb1 varex/1-471VHWTAEEKQLICSLWGKI--DVGLIGGETLAGLLVIYPWTQRQFSHFGNLSSPTAIAGNPRVKAHGKKVLTSFGDAIKNL
hbb2 xentr/1-471VHWTAEEKATIASVWGKV--DIEQDGHDALSRLLVVYPWTQRYFSSFGNLSNVSAVSGNVKVKAHGNKVLSAVGSAIQHL
hbb1 ranca/1-471VHWTAEEKAVINSVWQKV--DVEQDGHEALTRLFIVYPWTQRYFSTFGDLSSPAAIAGNPKVHAHGKKILGAIDNAIHNL
hbb2 tricr/1-471VHLTAEDRKEIAAILGKV--NVDSLGGQCLARLIVVNPWSRRYFHDFGDLSSCDAICRNPKVLAHGAKVMRSIVEATKHL
hba4 salir/1-471-SLSAKDKANVKAIWGKILPKSDEIGEQALSRMLVVYPQTKAYFSHWASVAP-----GSAPVKKHGITIMNQIDDCVGHM
myg_escgi/1-471 -VLSDAEWQLVLNIWAKVEADVAGHGQDILIRLFKGHPETLEKFDKFKHLKTEAEMKASEDLKKHGNTVLTALGGILKKK
```

Different positions are not equivalent

Sequence logos



http://weblogo.berkeley.edu/cache/file5h2DWc.png

The substitution score IN A FAMILY should depend on the position (the same for gaps)

For modelling families we need more flexible tools

Probabilistic Models for Biological Sequences

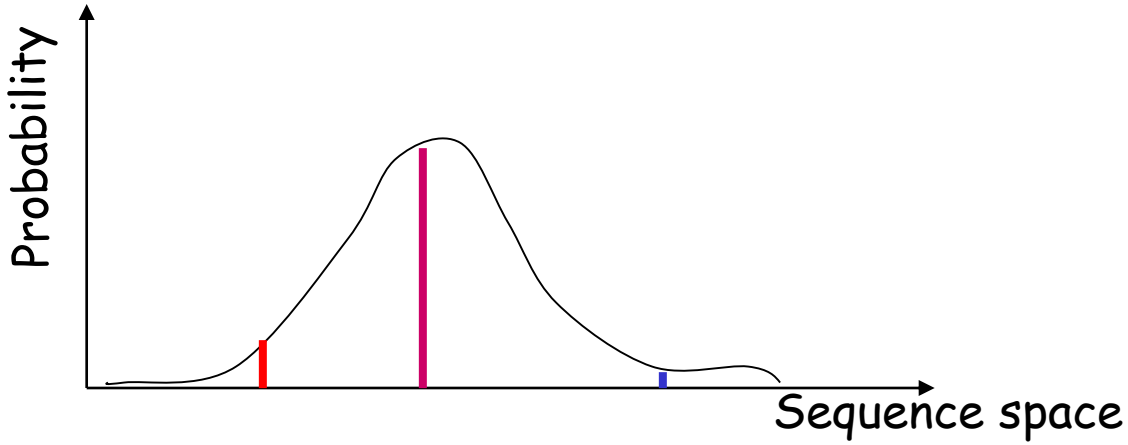
- What are they?

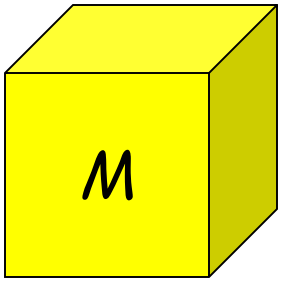
Probabilistic models for sequences

Generative definition:

- Objects producing different outcomes (sequences) with different probabilities

- The probability distribution over the sequences space determines the model specificity





Generates s_i with probability $P(s_i \mid M)$
e.g.: M is the representation of the family of globins

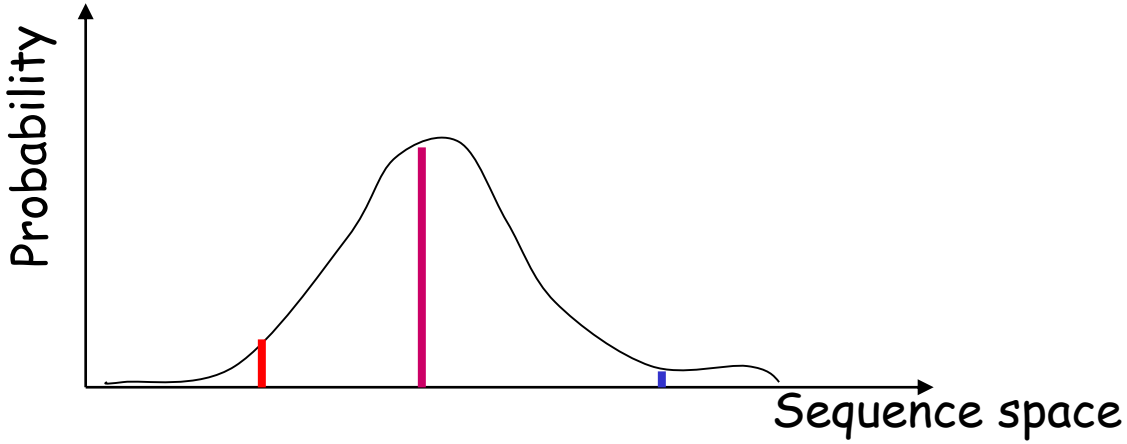
Probabilistic models for sequences

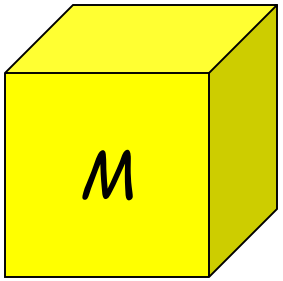
We don't need a generator of new biological sequences

the generative definition is useful as operative definition

Associative definition:

- Objects that, given an outcome (sequence), compute a probability value





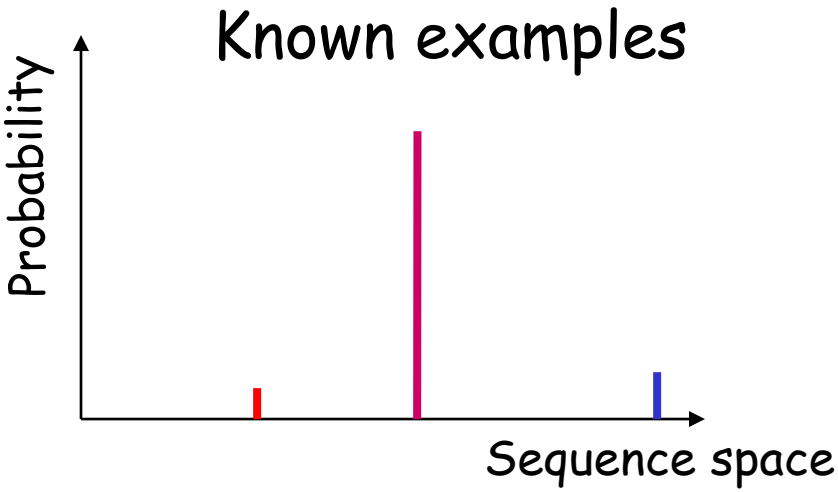
Associates probability $P(s_i \mid M)$ to s_i

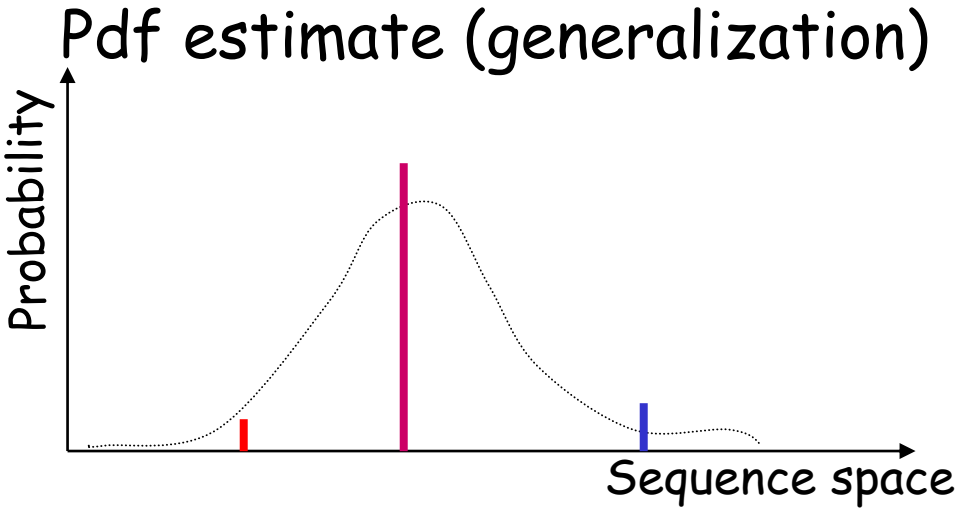
e.g.: M is the representation of the family of globins

Probabilistic models for sequences

Most useful probabilistic models are Trainable *systems*

The probability density function over the sequence space is estimated from known examples by means of a learning algorithm





e.g.: Writing a generic representation of the sequences of globins starting from a set of known globins

Probabilistic Models for Biological Sequences

- What are they?
- Why to use them?

Modelling a protein family

Probabilistic model

Given a protein class (e.g. Globins), a probabilistic model trained on this family can be adopted to compute a probability value for new sequences

Seq1	**0.98**
Seq2	**0.21**
Seq3	**0.12**
Seq4	**0.89**
Seq5	**0.47**
Seq6	**0.78**

This value measures the similarity between the new sequence and the family described by the model

Probabilistic Models for Biological Sequences

- What are they?
- Why to use them?
- Which probabilities do they compute?

$P(s \mid M)$ or $P(M \mid s)$?

A model M associates to a sequence s the probability $P(s \mid M)$

This probability answers the question:

Which is the probability for a model M (e.g. describing the Globins) to generate the sequence s ?

The question we want to answer is:

Given a sequence s, does it belong to the class described by the model M? (e.g. is it a Globin?)

We need to compute $P(M \mid s)$!!

Bayes Theorem

$$P(X,Y) = P(X \mid Y)\,P(Y) = P(Y \mid X)\,P(X)$$ Joint probability

$$\Downarrow$$

$$P(Y \mid X) = \frac{P(X \mid Y)\,P(Y)}{P(X)}$$

So:

$$P(M \mid s) = \frac{P(s \mid M)\,P(M)}{P(s)}$$

A priori probabilities

The *A priori* probabilities

$$P(M \mid s) = \frac{P(s \mid M)\,P(M)}{P(s)}$$

A priori probabilities

P(M) is the probability of the model (i.e. of the class described by the model) BEFORE we know the sequence:

can be estimated as the abundance of the class

P(s) is the probability of the sequence in the sequence space.

Cannot be reliably estimated!!

Comparison between models

We can overcome the problem comparing the probability of generating *s* from different models

$$\frac{P(M_1 \mid s)}{P(M_2 \mid s)} = \frac{P(s \mid M_1)\ P(M_1)}{P(s)} \frac{P(s)}{P(s \mid M_2)\ P(M_2)} =$$

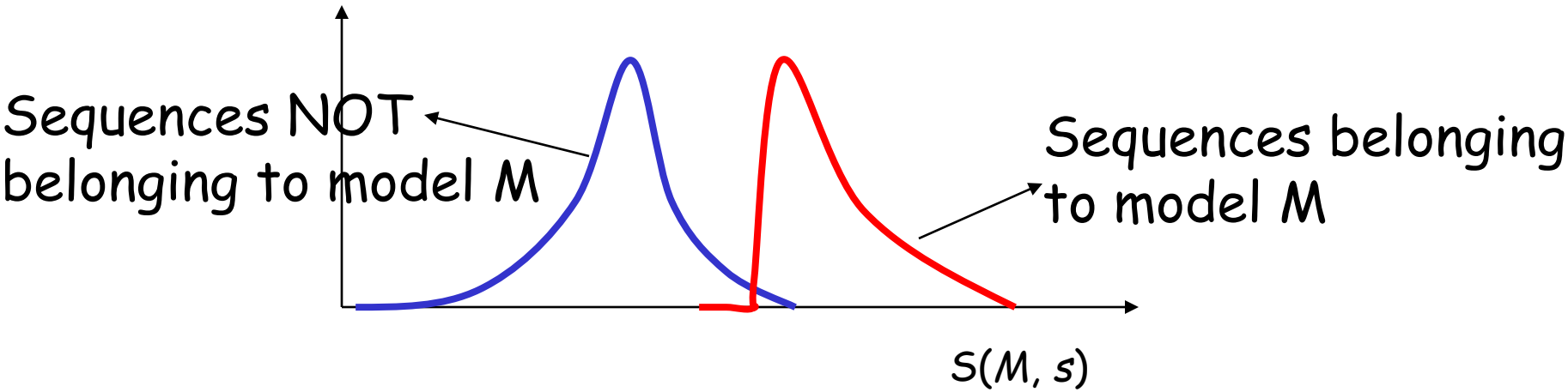
$$= \frac{P(s \mid M_1)}{P(s \mid M_2)} \frac{P(M_1)}{P(M_2)}$$

Ratio between the abundances of the classes

Null model

Alternatively, we can score a sequence for a model M comparing it to a Null Model:
a model that generates ALL the possible sequences with probabilities depending ONLY on letter (e.g. residue) statistical abundance

$$S(M, s) = \log \frac{P(s \mid M)}{P(s \mid N)}$$



In this case we need a threshold and a statistic for evaluating the significance (E-value, P-value)

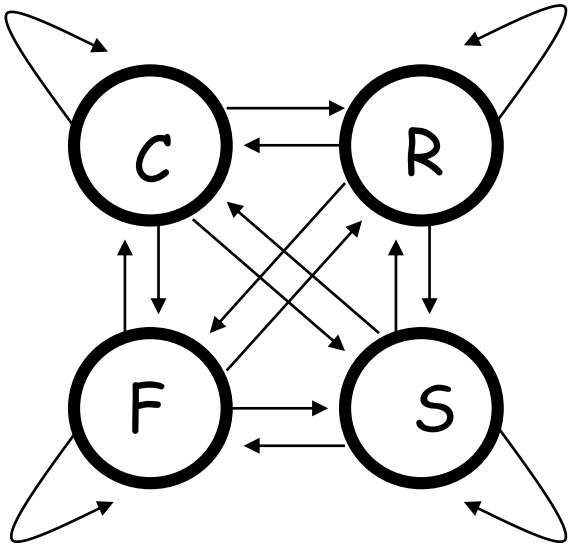
The simplest probabilistic models: Markov Models

- Definition

Markov Models Example: Weather

Register the weather conditions day by day:

as a first hypothesis the weather condition in a day probabilistically depends ONLY on the weather conditions in the day before .



Define the conditional probabilities

$P(C|C)$, $P(C|R)$,.... $P(R|C)$.....

C: Clouds
R: Rain
F: Fog
S: Sun

The probability for the 5-days registration CRRCS

$$P(CRRCS) = P(C)\cdot P(R|C)\cdot P(R|R)\cdot P(C|R)\cdot P(S|C)$$

Markov Model

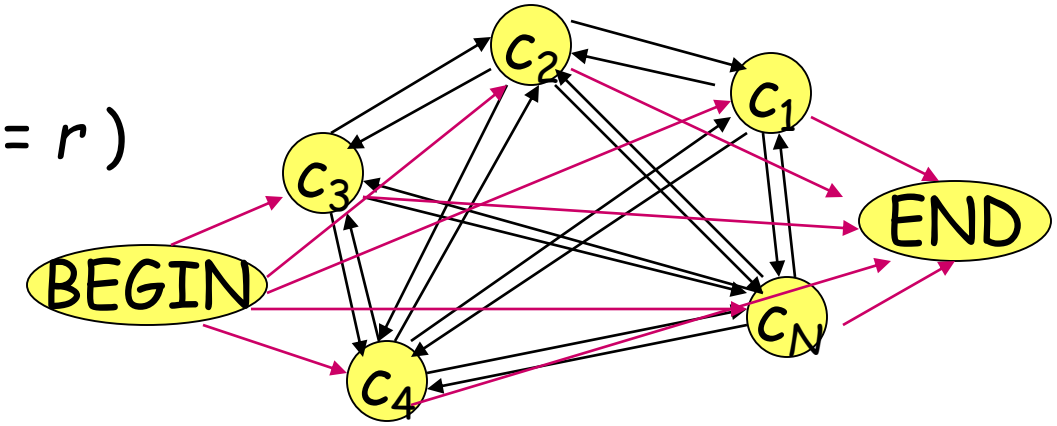
Stochastic generator of sequences in which the probability of state in position *i* depends ONLY on the state in position *i*-1

Given a set of states (== alphabet) $\mathcal{C} = \{c_1; c_2; c_3; \ldots\ldots\ldots c_N\}$

a Markov model is described with $N \times (N+2)$ parameters
$\{a_{rt}, a_{BEGIN\ t}, a_{r\ END}; r, t \in \mathcal{C}\}$

$a_{rq} = P(s^i = q \mid s^{i-1} = r)$
$a_{BEGIN\ q} = P(s^1 = q)$
$a_{r\ END} = P(s^T = END \mid s^{T-1} = r)$

$\sum_t a_{rt} + a_{r\ END} = 1 \quad \forall\ r$
$\sum_t a_{BEGIN\ t} = 1$



Markov Models

Given the sequence:

$$s = s^1 s^2 s^3 s^4 s^6 \ldots\ldots\ldots\ldots\ldots s^T$$

with $s^i \in \mathbf{C} = \{c_1; c_2; c_3; \ldots\ldots\ldots c_N\}$

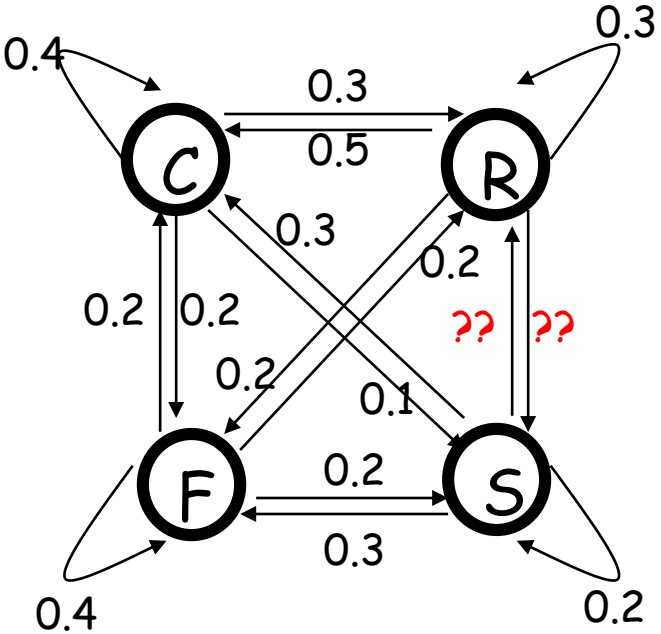
$$P(s \mid M) = P(s^1) \cdot \prod_{i=2} P(s^i \mid s^{i-1}) =$$

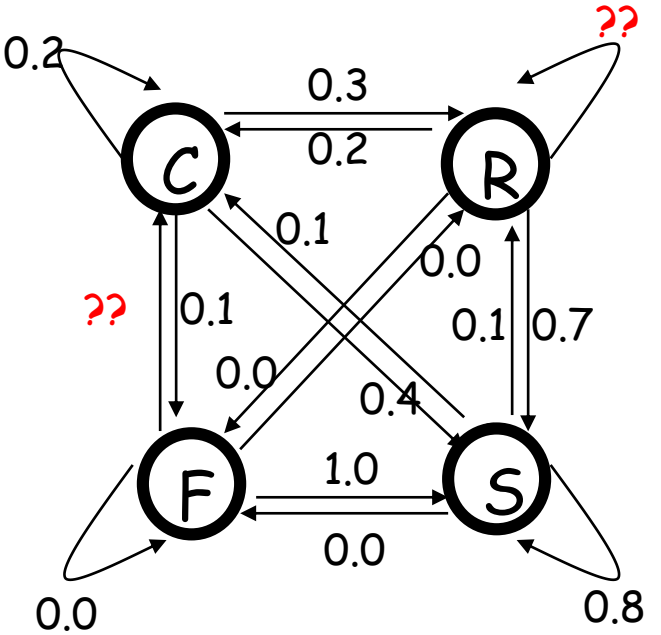
$$= a_{\text{BEGIN}\, s^1} \cdot \prod_{i=2}^{T} a_{s^{i-1} s^i} \cdot a_{s^T\, \text{END}}$$

$$P(\text{“ALKALI”}) = a_{\text{BEGIN}\, A} \cdot a_{A\,L} \cdot a_{L\,K} \cdot a_{K\,A} \cdot a_{A\,L} \cdot a_{L\,I} \cdot a_{I\,\text{END}}$$

Markov Models: Exercise

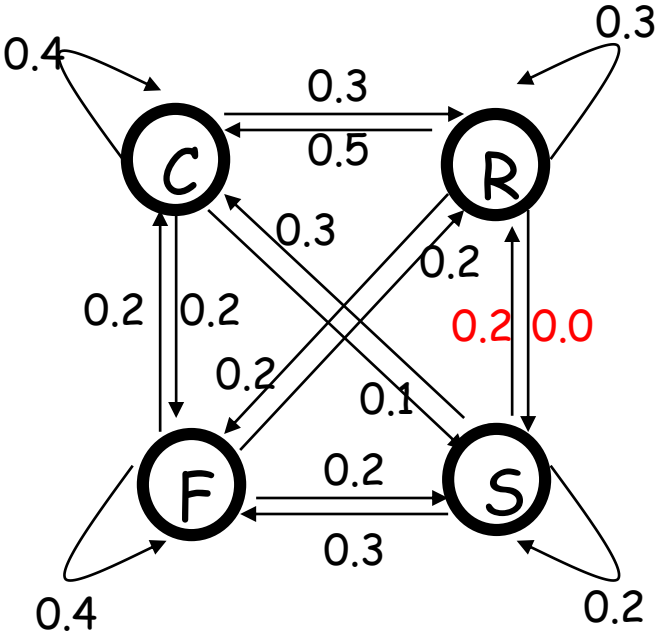
1) Fill the non defined values for the transition probabilities

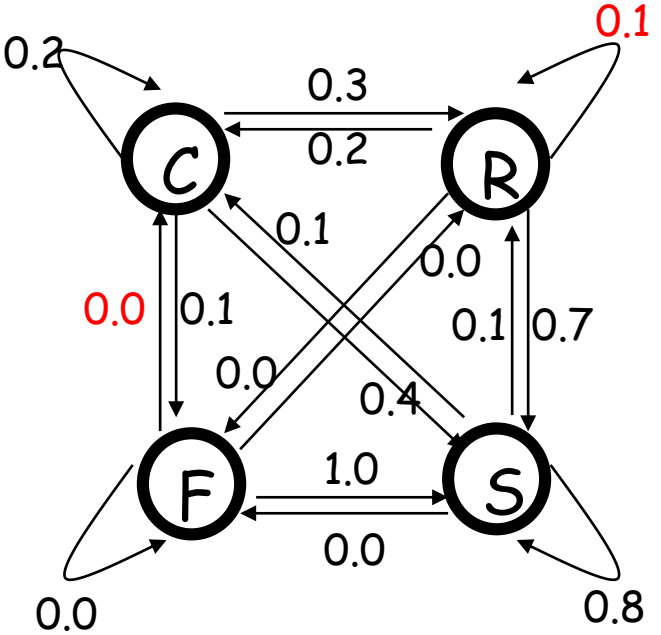




Markov Models: Exercise

2) Which model better describes the weather in summer? Which one better describes the weather in winter?

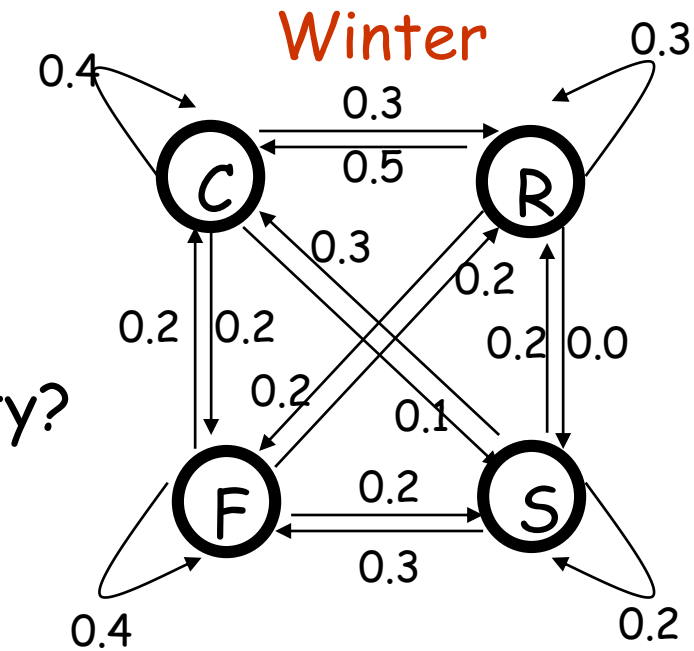


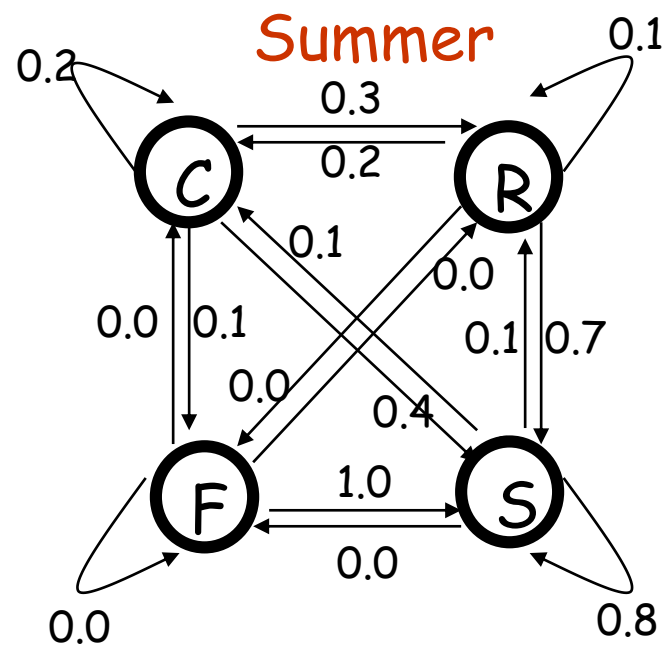


Markov Models: Exercise

3) Given the sequence
CSSSCFS

which model gives the higher probability?
[Consider the starting probabilities:
P(X|BEGIN)=0.25]



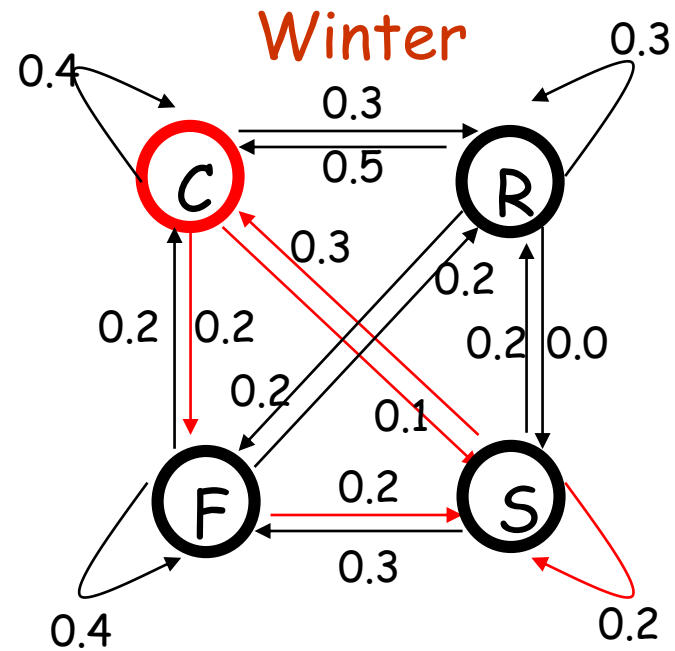


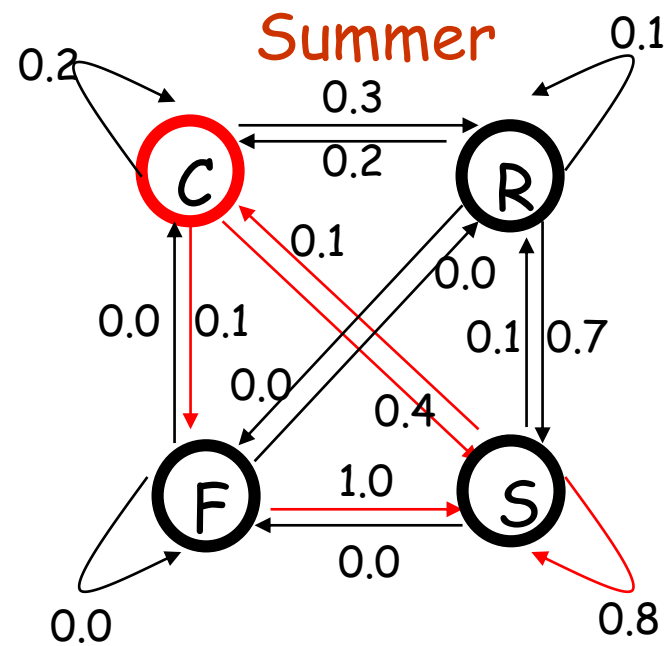
Markov Models: Exercise

P (CSSSCFS | Winter) =
=0.25x0.1x0.2x0.2x0.3x0.2x0.2=
=1.2 x 10^{-5}

P (CSSSCFS | Summer) =
=0.25x0.4x0.8x0.8x0.1x0.1x1.0=
=6.4 x 10^{-4}

4) Can we conclude that the observation sequence refers to a summer week?





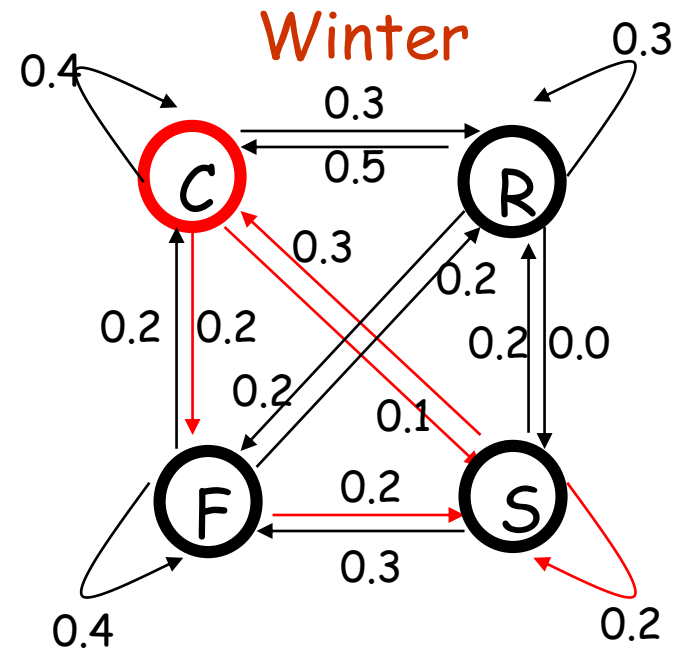
Markov Models: Exercise

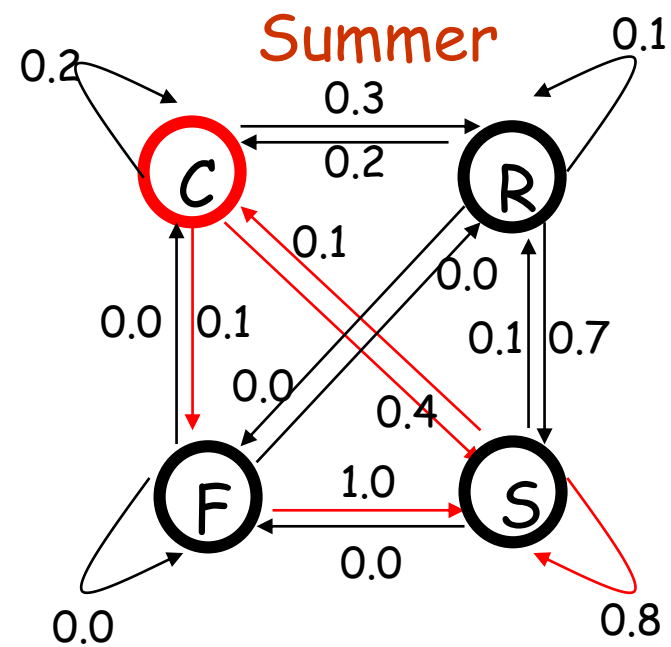
$P(\text{Seq} \mid \text{Winter}) = 1.2 \times 10^{-5}$

$P(\text{Seq} \mid \text{Summer}) = 6.4 \times 10^{-4}$

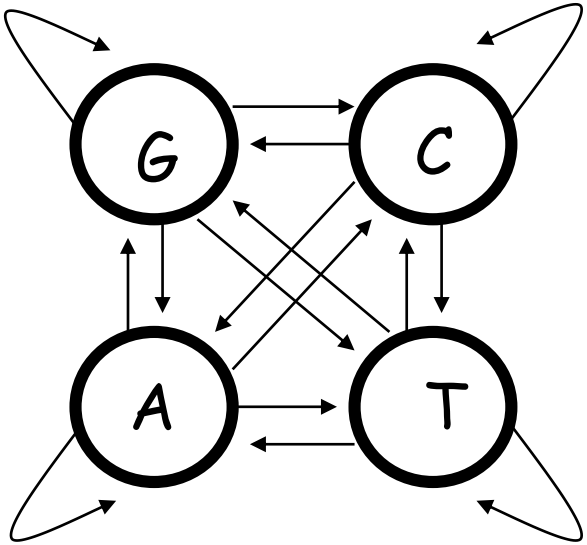
$$\frac{P(\text{Summer} \mid \text{Seq})}{P(\text{Winter} \mid \text{Seq})} =$$

$$= \frac{P(\text{Seq} \mid \text{Summer})\, P(\text{Summer})}{P(\text{Seq} \mid \text{Winter})\, P(\text{Winter})}$$





Simple Markov Model for DNA sequences



DNA:
$\mathcal{C}$ = {Adenine, Citosine, Guanine, Timine }

16 transition probabilities (12 of which independent) +
4 Begin probabilities +
4 End probabilities.

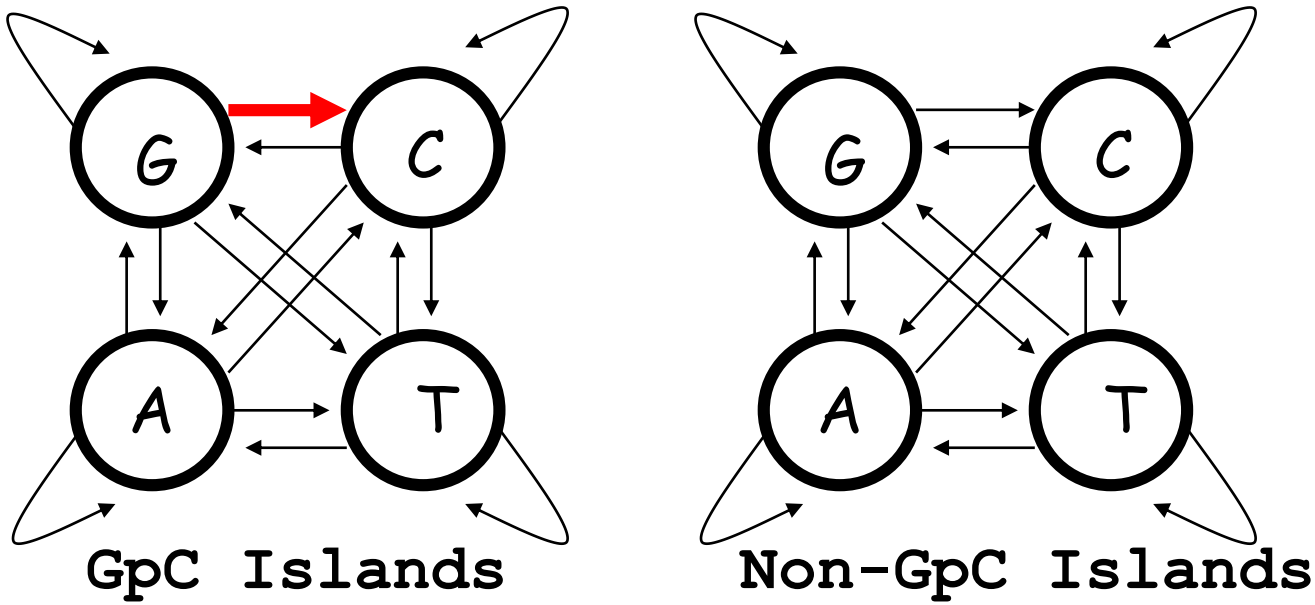
The parameters of the model are different in different zones of DNA

They describe the overall composition and the couple recurrences

Example of Markov Models: GpC Island

GATGCGTCGC

CTACGCAGCG



In the Markov Model of GpC Islands a_{GC} is higher than in Markov Model Non-GpC Islands

Given a sequence *s* we can evaluate

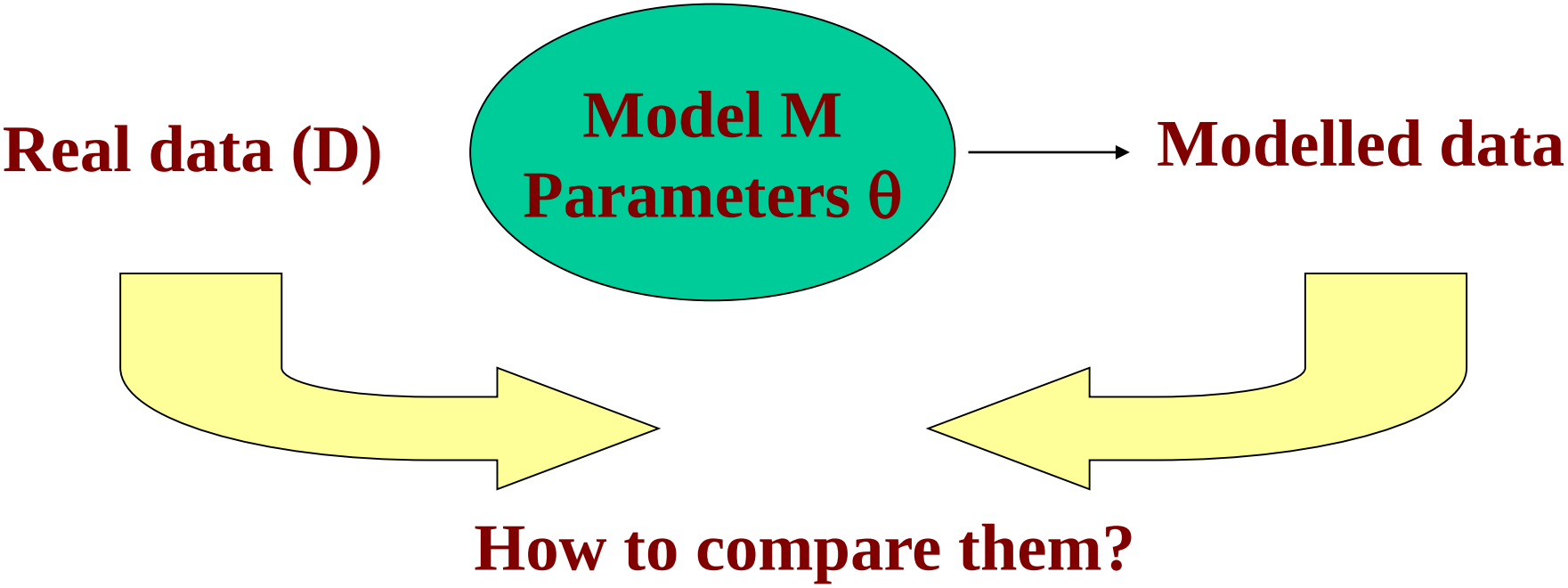
$$P(GpC \mid s) = \frac{P(s \mid GpC) \cdot P(GpC)}{P(s \mid GpC) \cdot P(GpC) + P(s \mid nonGpC) \cdot P(nonGpC)}$$

The simplest probabilistic models: Markov Models

- Definition
- Training

Probabilistic training of a parametric method

Generally speaking, a parametric model M aims to reproduce a set of known data



Training of Markov Models

Let θ_M be the set of parameters of model M.

During the training phase, θ_M parameters are estimated from the set of known data $\boldsymbol{D}$

Maximum Likelihood Extimation (ML)

$$\theta^{ML} = argmax_{\theta}\, P(\, D \mid M,\, \theta\,)$$

Maximum Likelihood training: Proof

Given a sequence s contained in **D**:

$$s = s^1 s^2 s^3 s^4 s^6 \ldots\ldots\ldots\ldots\ldots s^T$$

$$P(s \mid M) = a_{BEGIN,s^1} \cdot \prod_{i=2}^{T-1} a_{s^i s^{i+1}} \cdot a_{s^T END}$$

We can count the number of transitions between any to states j and k: n_{jk}

$$P(s \mid M) = \prod_{j=0}^{N+1} \prod_{k=0}^{N+1} a_{jk}^{\;n_{jk}}$$

Where states 0 and N+1 are BEGIN and END

On top of this, keep in mind that normalisation contstraints must be satisfied for each state

$$\forall j : \sum_{k'=0}^{N} a_{jk'} = 1$$

So the likelihood has to be maximised on the variety defined by the normalisation contraints. How to do that?

Maximum Likelihood training: Proof

Given a sequence s contained in **D**:

$$s = s^1 s^2 s^3 s^4 s^6 \ldots\ldots\ldots\ldots\ldots s^T$$

$$P(s \mid M) = a_{BEGIN,s^1} \cdot \prod_{i=2}^{T-1} a_{s^i s^{i+1}} \cdot a_{s^T END}$$

We can count the number of transitions between any to states j and k: n_{jk}

$$P(s \mid M) = \prod_{j=0}^{N+1} \prod_{k=0}^{N+1} a_{jk}^{\,n_{jk}}$$ Where states 0 and N+1 are BEGIN and END

Normalisation contstraints are taken into account using the Lagrange multipliers λ_k

$$L\left(\alpha_{ab}, \lambda_1, \lambda_{2,} .. \lambda_N\right) = P(s \mid M) - \sum_{j=0}^{N} \lambda_j \cdot \left(\sum_{k=0}^{N} a_{jk} - 1 \right)$$

$$\begin{cases} \dfrac{\partial L}{\partial a_{jk}} = \dfrac{n_{jk}}{a_{jk}} P(s \mid M) - \lambda_j = 0 \\ \dfrac{\partial L}{\partial \lambda_j} = \sum_{k'=0}^{N} a_{jk'} - 1 = 0 \end{cases} \Rightarrow \quad a_{jk} = \frac{n_{jk}}{\sum_{k'=0}^{N} n_{jk'}}$$

Training of Markov Models

Let θ_M be the set of parameters of model M.

During the training phase, θ_M parameters are estimated from the set of known data $\boldsymbol{D}$

Maximum Likelihood Extimation (ML)

$$\theta^{ML} = argmax_{\theta}\, P(\, D \mid M,\, \theta\,)$$

It can be proved that:

$$a_{ik} = \frac{n_{ik}}{\sum_j n_{ij}}$$

Frequency of occurrence as counted in the data set $\boldsymbol{D}$

Maximum A Posteriori Extimation (MAP)

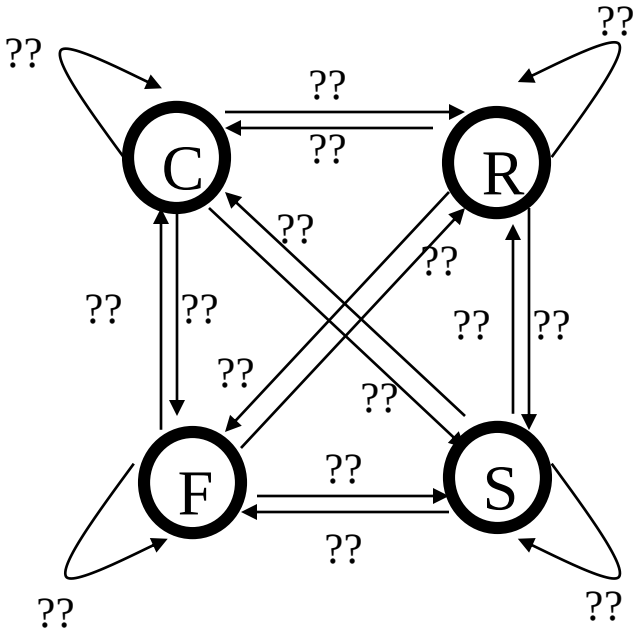
$$\theta^{MAP} = argmax_{\theta}\, P(\, \theta \mid M, D\,) = argmax_{\theta}\, [P(\, D \mid M,\, \theta\,) \cdot P(\theta\,)\,]$$

Training of Markov Models: Exercise

Given the observation sequence:

CCCFFCRRRCCSSSSFSFRRFFSSF

set the parameters of the Markov Model.



Hidden Markov Models

- Preliminary examples

Loaded dice

We have 99 regular dice (***R***) and 1 loaded die (***L***).

	P(1)	**P(2)**	**P(3)**	**P(4)**	**P(5)**	**P(6)**
R	1/6	1/6	1/6	1/6	1/6	1/6
L	1/10	1/10	1/10	1/10	1/10	1/2

Given a sequence:

`415626665632163654366215261153626416236426166461626 3`

We <u>don't know</u> the sequence of dice that generated it.

`RRRRRLRLRRRRRRRLRRRRRRRRRRRRLRLRRRRRRRRLRRRRLRRRRLRR`

Loaded dice

Hypothesis:

We chose a different die for each roll

Two stochastic processes give origin to the sequence of observations.

1) Choosing the die (***R*** o ***L***).

2) Rolling the die

The sequence of dice is *hidden*

The first process is assumed to be Markovian (in this case a 0-order MM)

The outcome of the second process depends only on the state reached in the first process (that is the chosen die)

Casinò

Model

Each state (***R*** and ***L***) generates a character of the alphabet
C = {1, 2, 3, 4, 5, 6 }

The *emission probabilities* depend only on the state.

The *transition probabilities* describe a Markov model that generates a state path: the *hidden sequence* (π)

The *observations sequence* (*s*) is generated by two concomitant stochastic processes

The *observations sequence* (*s*) is generated by two concomitant stochastic processes

4156266656321636543662152 61
RRRRRLRLRRRRRRRLRRRRRRRRRRR

Choose the State : R **Probability= 0.99**

Chose the Symbol: 1 **Probability= 1/6 (given R)**

4156266656321636543662152611
RRRRRLRLRRRRRRRLRRRRRRRRRRRR

The *observations sequence* (*s*) is generated by two concomitant stochastic processes

415626665632163654366215261 1
RRRRRLRLRRRRRRRLRRRRRRRRRRR

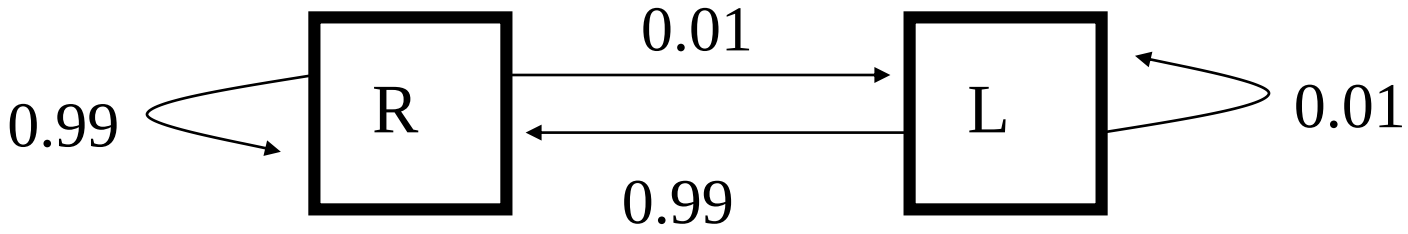
Choose the State : L **Probability= 0.99**

Chose the Symbol: 5 **Probability= 1/10 (given L)**

4156266656321636543662152611 5
RRRRRLRLRRRRRRRLRRRRRRRRRRRL

Loaded dice

Model



Each state (***R*** and ***L***) generates a character of the alphabet

$$\boldsymbol{C} = \{1, 2, 3, 4, 5, 6\}$$

The *emission probabilities* depend only on the state.

The *transition probabilities* describe a Markov model that generates a state path: the <u>*hidden sequence*</u> (π)

The *observations sequence* (s) is generated by two concomitant stochastic processes

Why to search for the hidden path?
Some not so serious example

1) DEMOGRAPHY

Observable: Number of births and deaths in a year in a village.
Hidden variable: Economic conditions (as a first approximation we can consider the success in business as a random variable, and by consequence, the wealth as a Markov variable

---> can we deduce the economic conditions of a village during a century by means of the register of births and deaths?

2) THE METEREOPATHIC TEACHER

Observable: Average of the marks that a meteoropathic teacher gives to their students during a day.
Hidden variable: Weather conditions

---> can we deduce the weather conditions during a years by means of the class register?

Why to search for the hidden path?
To be more serious

1) SECONDARY STRUCTURE

Observable: protein sequence

Hidden variable: secondary structure

---> can we deduce (predict) the secondary structure of a protein given its amino acid sequence?

2) ALIGNMENT

Observable: protein sequence

Hidden variable: position of each residue along the alignment of a protein family

---> can we align a protein to a family, starting from its amino acid sequence?

Hidden Markov Models

- Preliminary examples
- Formal definition

Formal definition of Hidden Markov Models

A HMM is a stochastic generator of sequences characterised by:

- N states
- A set of transition probabilities between two states $\{\boldsymbol{a_{kj}}\}$

  $\boldsymbol{a_{kj}} = P(\pi(i) = \boldsymbol{j} \mid \pi(i\text{-}1) = \boldsymbol{k})$
- A set of starting probabilities $\{\boldsymbol{a_{0k}}\}$

  $\boldsymbol{a_{0k}} = P(\pi(1) = \boldsymbol{k})$
- A set of ending probabilities $\{\boldsymbol{a_{k0}}\}$

  $\boldsymbol{a_{k0}} = P(\pi(i) = \text{END} \mid \pi(i\text{-}1) = \boldsymbol{k})$
- An alphabet $\boldsymbol{C}$ with M characters.
- A set of emission probabilities for each state $\{\boldsymbol{e_k(c)}\}$

  $\boldsymbol{e_k(c)} = P(s^i = \boldsymbol{c} \mid \pi(i) = \boldsymbol{k})$
- Constraints:

  $\Sigma_{\boldsymbol{k}}\ \boldsymbol{a_{0k}} = 1$

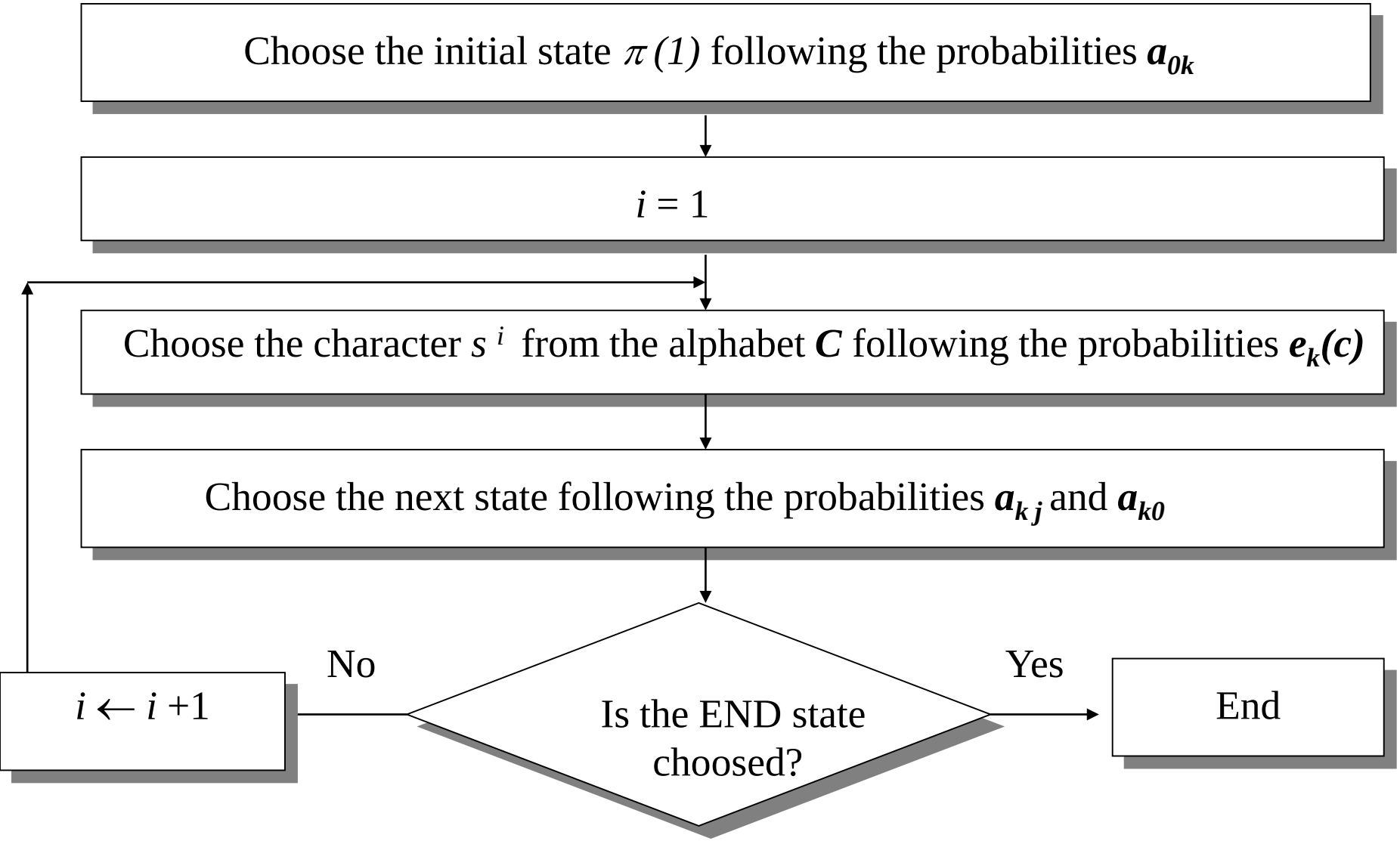
  $\boldsymbol{a_{k0}} + \Sigma_{\boldsymbol{j}}\ \boldsymbol{a_{kj}} = 1 \qquad \forall\ \boldsymbol{k}$

  $\Sigma_{\boldsymbol{c} \in \boldsymbol{C}}\ \boldsymbol{e_k(c)} = 1 \qquad \forall\ \boldsymbol{k}$

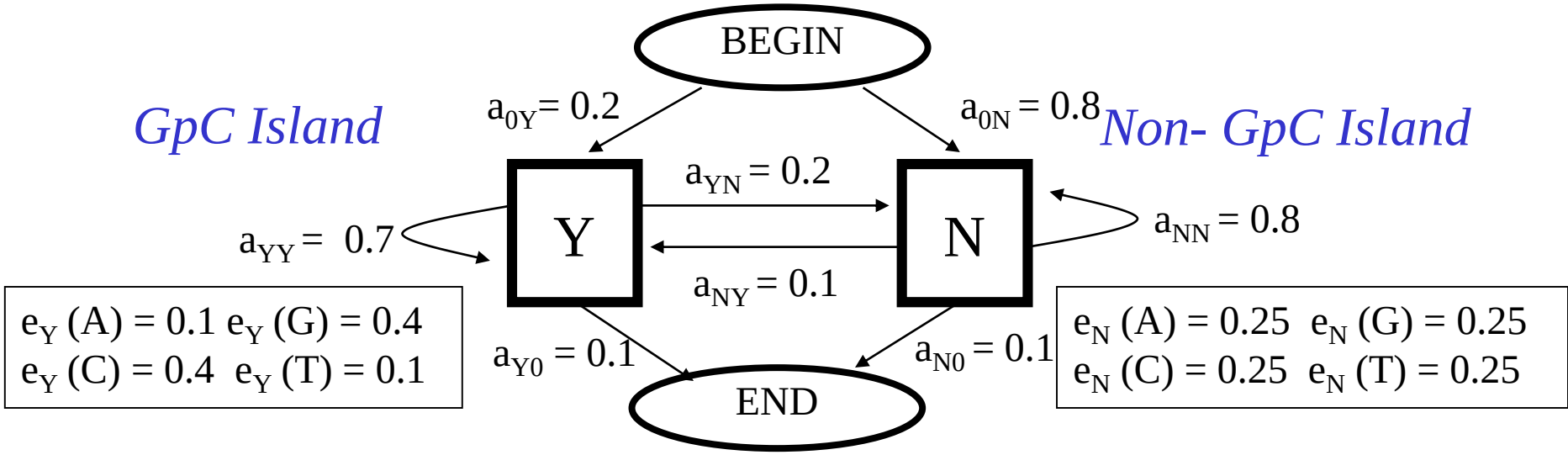
s: sequence

π: path through the states

Generating a sequence with a HMM



GpC Island, simple model

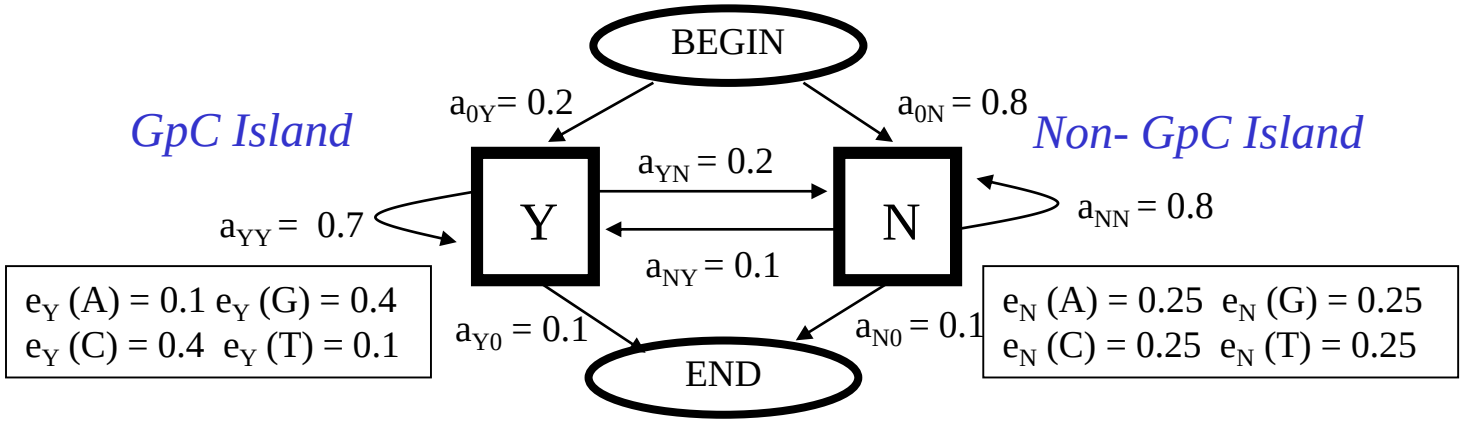


s :**AGCGCGTAATCTG**

π :**YYYYYYYNNNNNN**

- ***P(s, π | M) can be easily computed***

P(s, π | M) can be easily computed



s :	**A**	**G**	**C**	**G**	**C**	**G**	**T**	**A**	**A**	**T**	**C**	**T**	**G**
π :	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**N**	**N**	**N**	**N**	**N**	**N**

Emission: $0.1 \times 0.4 \times 0.4 \times 0.4 \times 0.4 \times 0.4 \times 0.1 \times 0.25 \times 0.25 \times 0.25 \times 0.25 \times 0.25 \times 0.25$

Transition: $0.2 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.2 \times 0.8 \times 0.8 \times 0.8 \times 0.8 \times 0.8 \times 0.1$

Multiplying all the probabilities gives the probability of having the sequence AND the path through the states

Evaluation of the joint probability of the sequence ad the path

$$P(s,\pi \mid M) = P(s \mid \pi, M) \cdot P(\pi \mid M)$$

$$P(\pi \mid M) = a_{0\pi(1)} \cdot \prod_{i=2}^{T} a_{\pi(i-1)\pi(i)} \cdot a_{\pi(T)0}$$

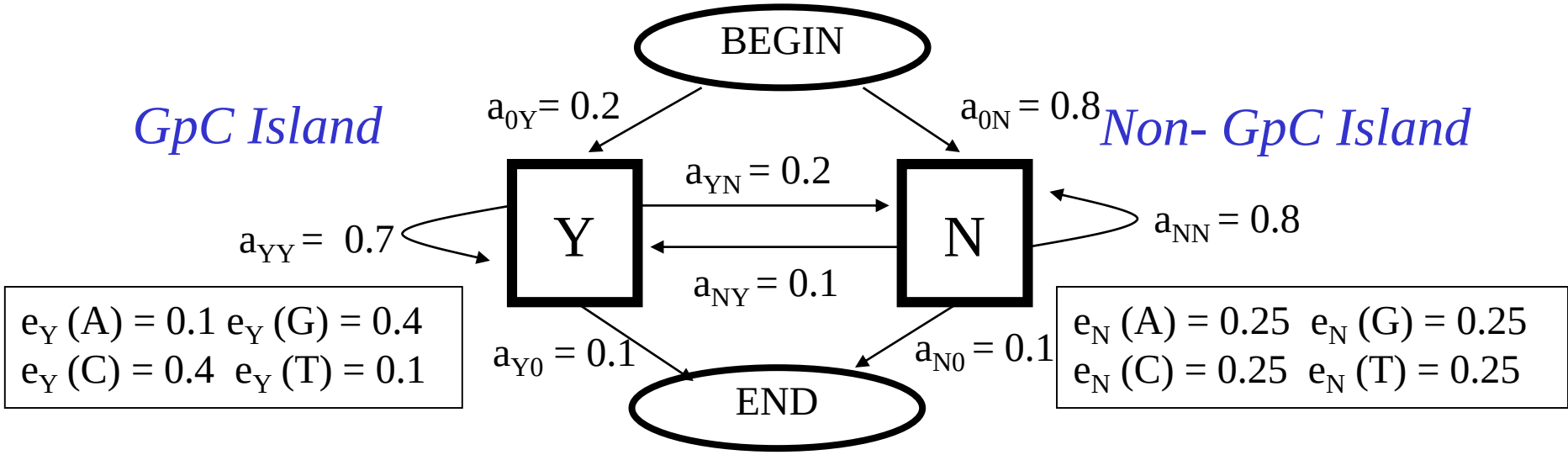
$$P(s \mid \pi, M) = \prod_{i=1}^{T} e_{\pi(i)}(s^i)$$

$$P(s,\pi \mid M) = a_{\pi(T)0} \cdot \prod_{i=1}^{T} a_{\pi(i-1)\pi(i)} \cdot e_{\pi(i)}(s^i)$$

Hidden Markov Models

- Preliminary examples
- Formal definition
- Three questions

GpC Island, simple model

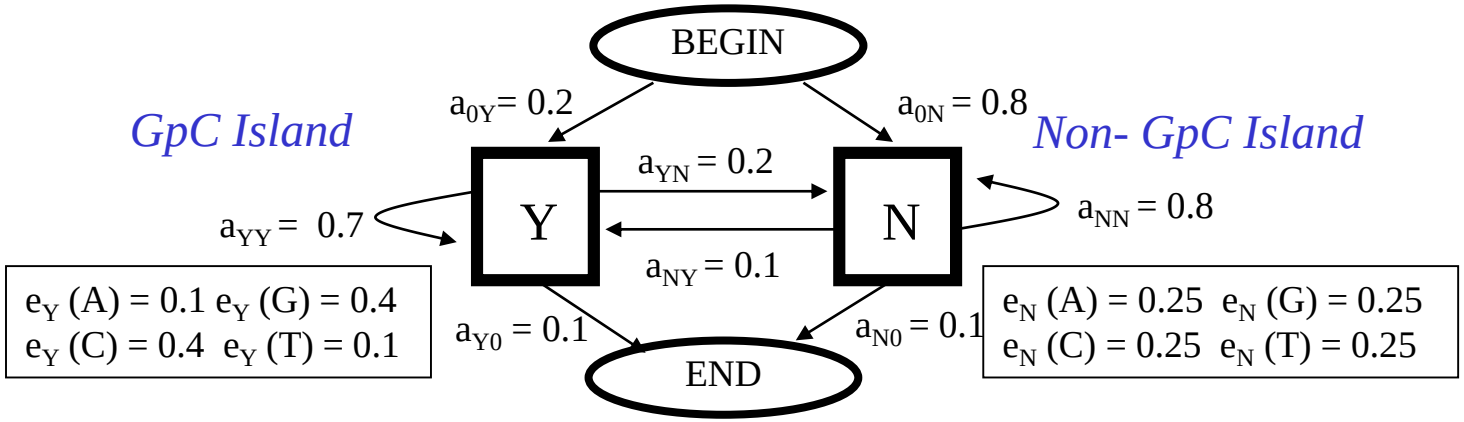


s :**AGCGCGTAATCTG**

π:**????????????**

- ***P(s, π | M) can be easily computed***
- ***How to evaluate P (s | M), when the path is unknown?***

How to evaluate P (s | M)?



$$P(s \mid M) = \Sigma_\pi P(s, \pi \mid M)$$

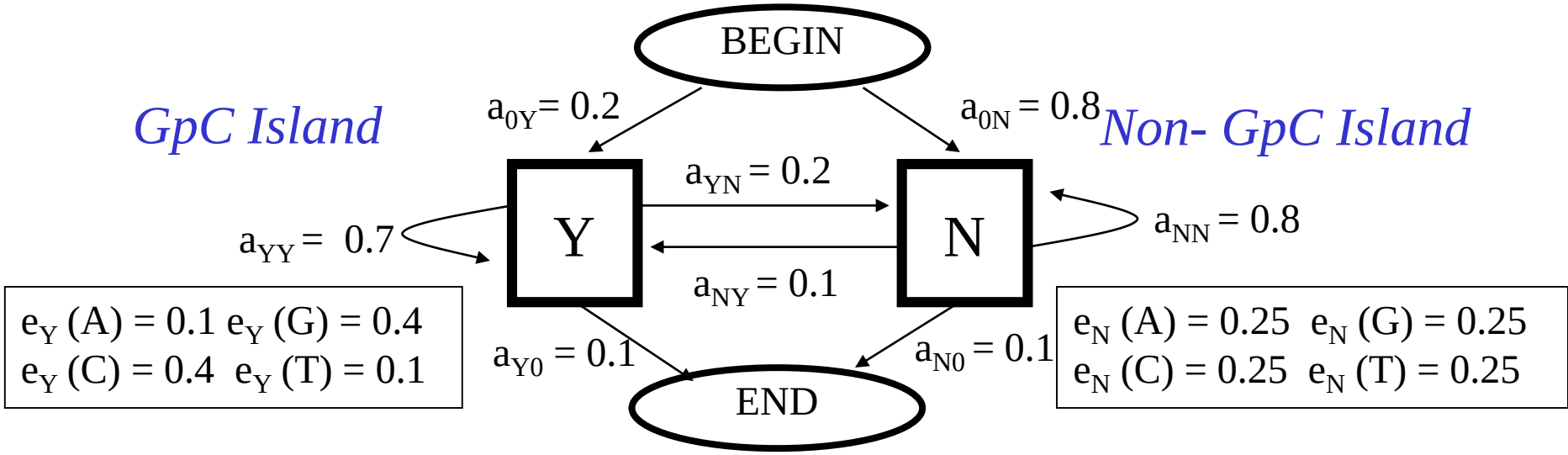
s :	A	G	C	G	C	G	T	A	A	T	C	T	G
π_1:	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
π_2:	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
π_3:	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y
π_4:	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N
π_5:	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y

.....

2^{13} different paths

Summing over all the path will give the probability of having the sequence

Resumé: GpC Island, simple model

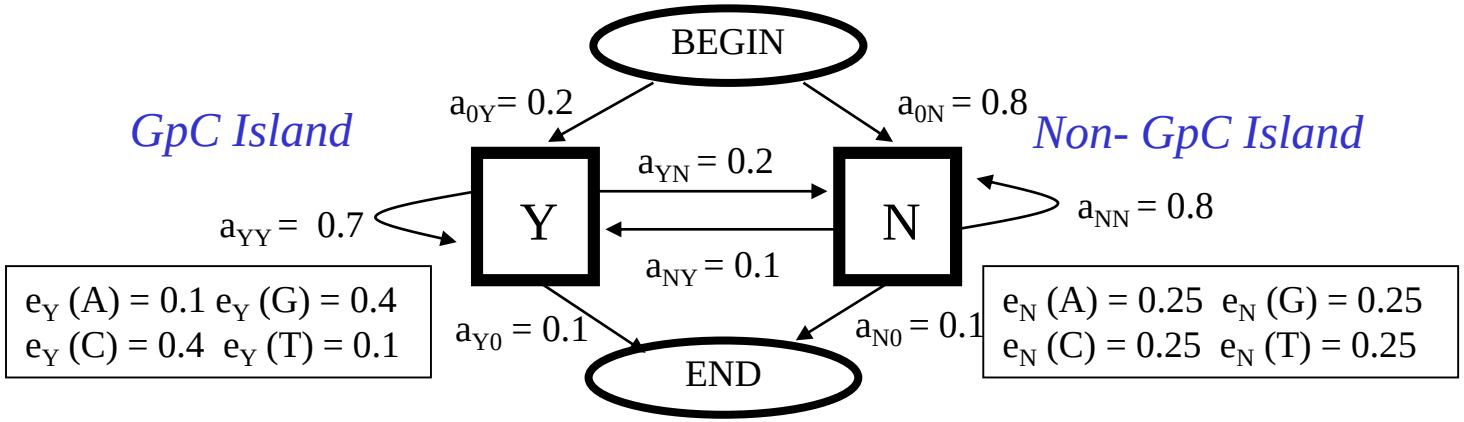


s :**AGCGCGTAATCTG**

π :**????????????**

- ***P(s, π | M) can be easily computed***
- ***How to evaluate P (s | M), when the path is unknown?***
- ***Can we show the hidden path?***

Can we show the hidden path?



$$\pi^* = \text{argmax}_{\pi} [\, P(\, \pi \mid s, M\,)\,] = \text{argmax}_{\pi} [\, P(\, \pi\,, s \mid M\,)\,]$$

s :	A	G	C	G	C	G	T	A	A	T	C	T	G
π_1:	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
π_2:	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N
π_3:	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y
π_4:	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N
π_5:	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y

.....

2^{13} different paths

Viterbi path: path that gives the best joint probability

Can we show the hidden path?

Viterbi decoding

Among all the possible path, choose the path $\pi*$ that maximises the $P(\pi \mid s, M)$

$$\pi^* = \text{argmax}_{\pi} [\, P(\pi \mid s, M) \,] = \text{argmax}_{\pi} [\, P(\pi, s \mid M) \,]$$

A Posteriori decoding

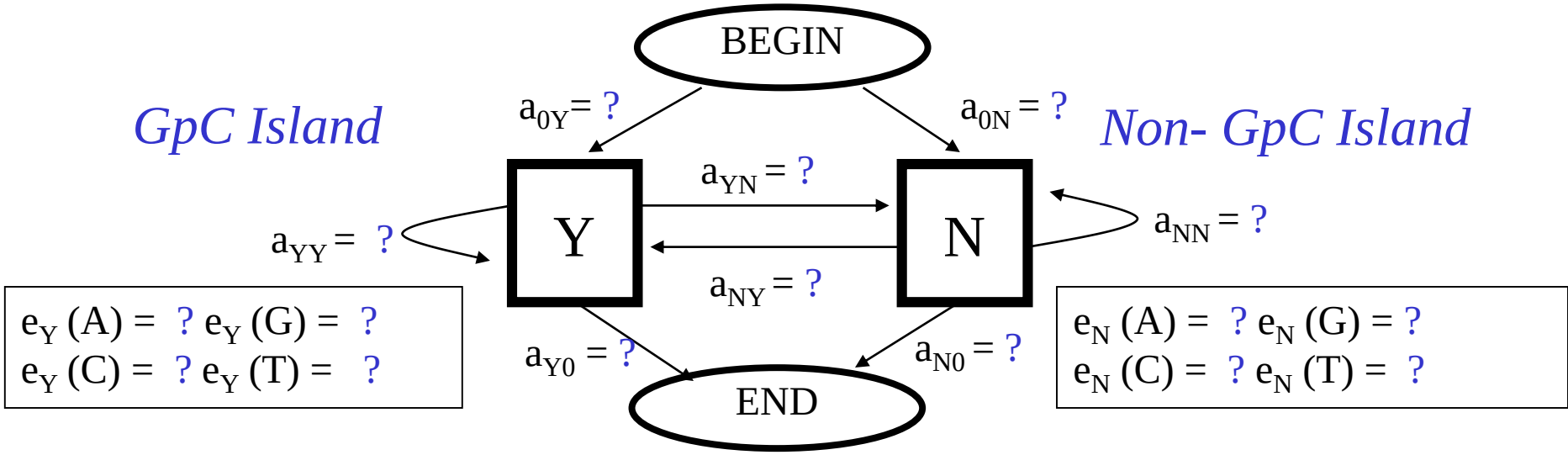
For each position choose the state $\underline{\pi}(t)$:

$$\underline{\pi}(i) = \text{argmax}_{k} [\, P(\pi(i) = k \mid s, M) \,]$$

The contribution to this probability derives from all the paths that go through the state k at position i.

The A posteriori path can be a non-sense path (it may not be a legitimate path if some transitions are not permitted in the model)

GpC Island, simple model

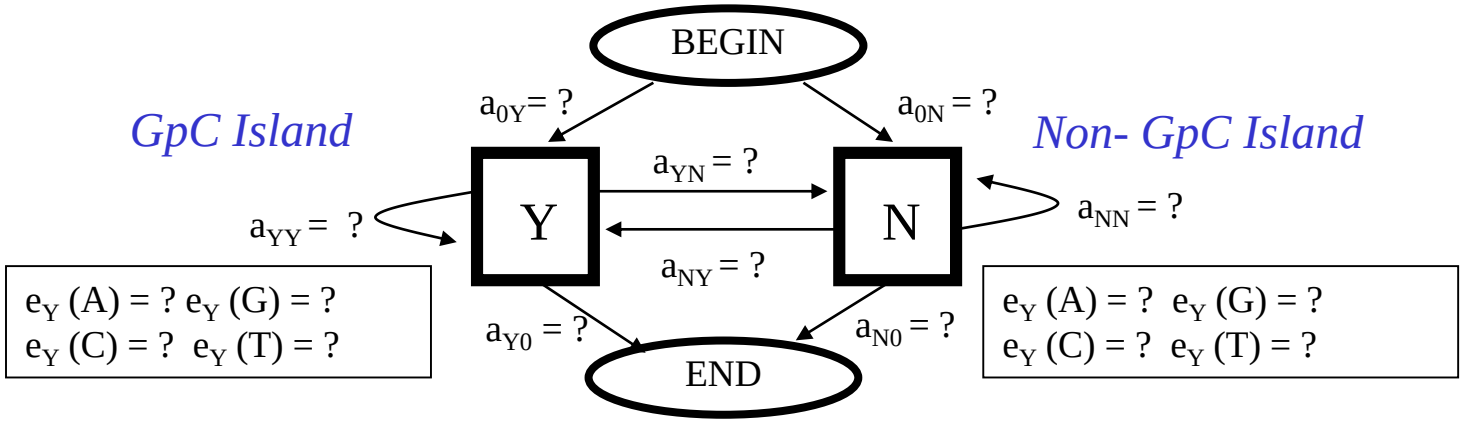


s :**AGCGCGTAATCTG**

π :**YYYYYYYNNNNNN**

- ***P(s, π | M) can be easily computed***
- ***How to evaluate P (s | M), when the path is unknown?***
- ***Can we show the hidden path?***
- ***Can we evaluate the parameters starting from known examples?***

Can we evaluate the parameters starting from known examples?



s :	**A**	**G**	**C**	**G**	**C**	**G**	**T**	**A**	**A**	**T**	**C**	**T**	**G**
π :	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**N**	**N**	**N**	**N**	**N**	**N**

Emission: $e_Y(A) \times e_Y(G) \times e_Y(C) \times e_Y(G) \times e_Y(C) \times e_Y(G) \times e_Y(T) \times e_N(A) \times e_N(A) \times e_N(T) \times e_N(C) \times e_N(T) \times e_N(G)$

Transition: $a_{0Y} \times a_{YY} \times a_{YY} \times a_{YY} \times a_{YY} \times a_{YY} \times a_{YY} \times a_{YN} \times a_{NN} \times a_{NN} \times a_{NN} \times a_{NN} \times a_{NN} \times a_{N0}$

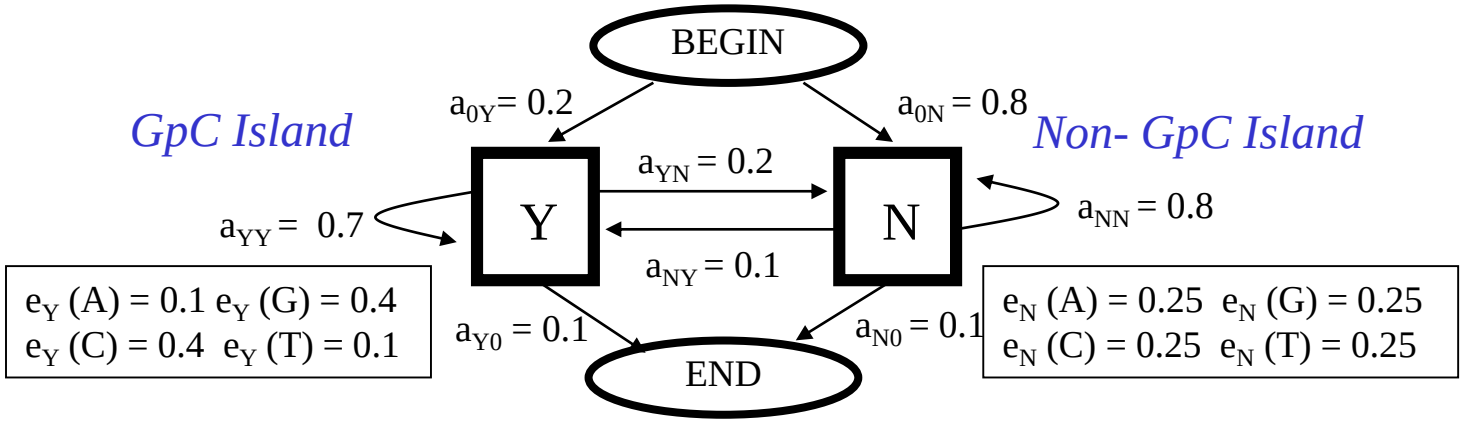
How to find the parameters *e* and *a* that maximises this probability?

How if we don't know the path?

Hidden Markov Models: Algorithms

- Resumé
- Evaluating $P(s \mid M)$: Forward Algorithm

Computing P(s,π | M) for each path is a redundant operation



s :	**A**	**G**	**C**	**G**	**C**	**G**	**T**	**A**	**A**	**T**	**C**	**T**	**G**
π :	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**

Emission: $0.1 \times 0.4 \times 0.4 \times 0.4 \times 0.4 \times 0.4 \times 0.1 \times 0.1 \times 0.1 \times 0.1 \times 0.4 \times 0.1 \times 0.4$

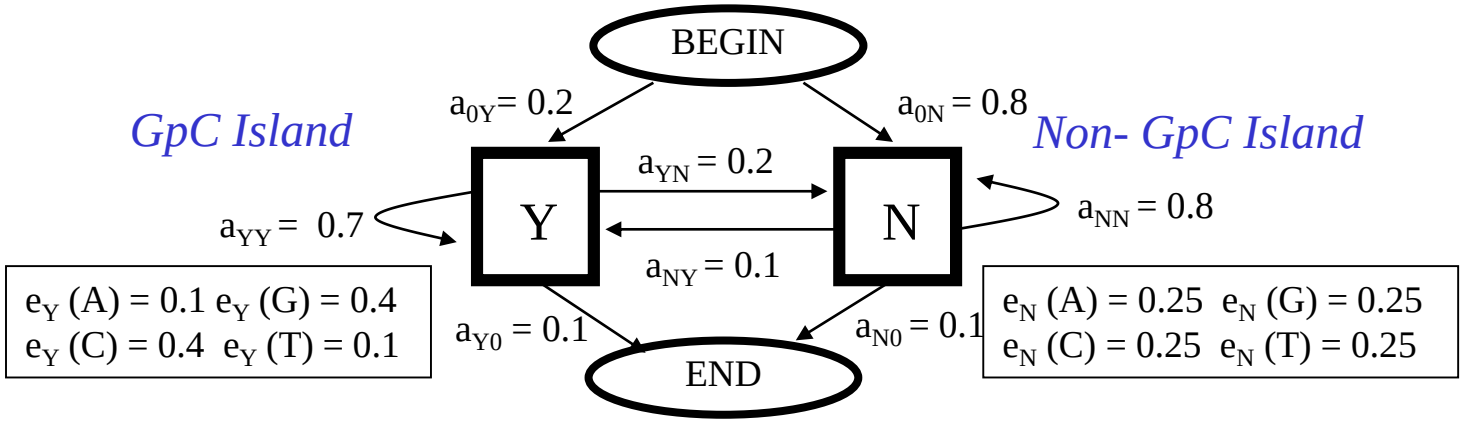
Transition: $0.2 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.1$

s :	**A**	**G**	**C**	**G**	**C**	**G**	**T**	**A**	**A**	**T**	**C**	**T**	**G**
π :	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**N**

Emission: $0.1 \times 0.4 \times 0.4 \times 0.4 \times 0.4 \times 0.4 \times 0.1 \times 0.1 \times 0.1 \times 0.1 \times 0.4 \times 0.1 \times 0.25$

Transition: $0.2 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.2 \times 0.1$

Strategy: Summing over all the possible paths



s : **A** **G**

π : **Y** **Y**

Emission: 0.1×0.4

Transition: 0.2×0.7 *0.0056*

s : **A** **G**

π : **N** **Y**

Emission: 0.25×0.4

Transition: 0.8×0.1 *0.008*

---

s : **A** **G**

π : **Y** **N**

Emission: 0.1×0.25

Transition: 0.2×0.2 *0.001*

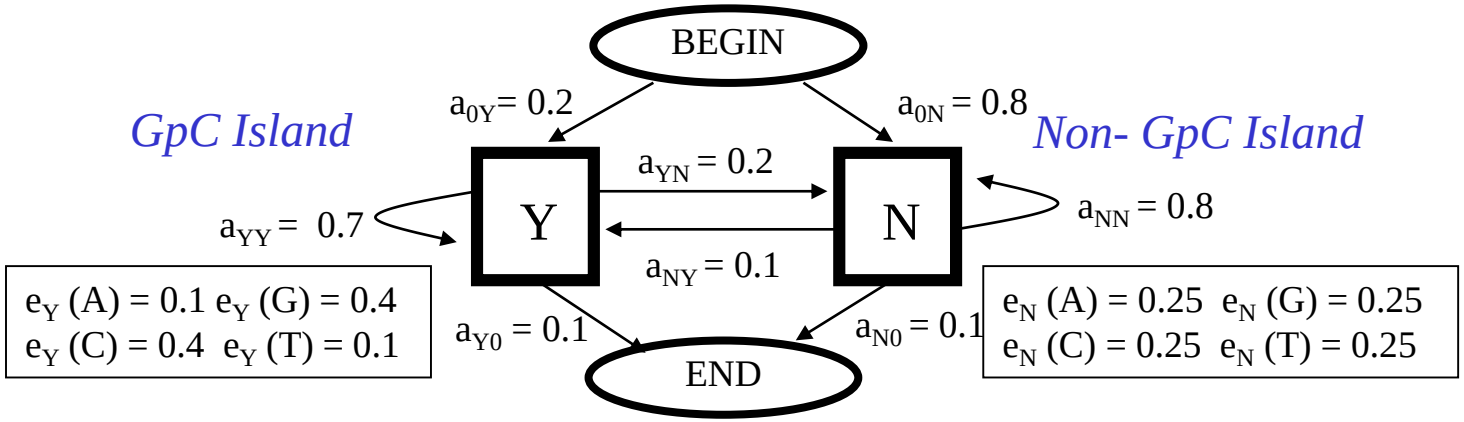
s : **A** **G**

π : **N** **N**

Emission: 0.25×0.25

Transition: 0.8×0.8 *0.04*

Strategy: Summing over all the possible paths



s : **A** **G**

π : **Y** **Y**

Emission: 0.1×0.4

Transition: 0.2×0.7 *0.0056*

s : **A** **G**

π : **N** **Y**

Emission: 0.25×0.4

Transition: 0.8×0.1 *0.008*

In order to continue the process we only need to know the last state:

The contributions from previous states can be summed up

s : **A** **G**

π : **Y** **N**

Emission: 0.1×0.25

Transition: 0.2×0.2 *0.001*

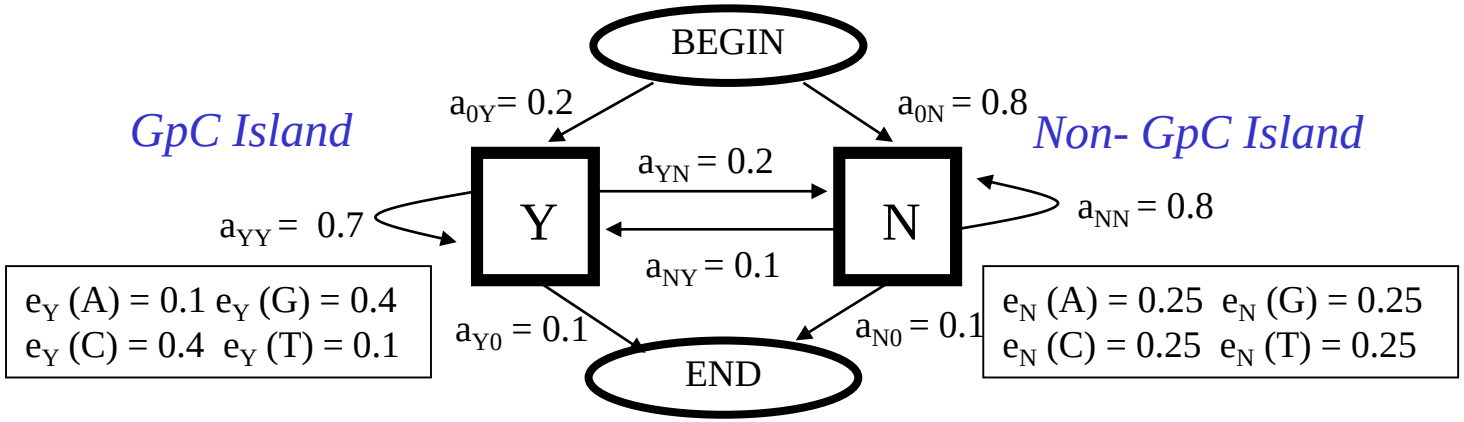
s : **A** **G**

π : **N** **N**

Emission: 0.25×0.25

Transition: 0.8×0.8 *0.04*

Strategy: Summing over all the possible paths



s : **A** **G**
π : **Y** **Y**

Emission: 0.1×0.4
Transition: 0.2×0.7 *0.0056*

s : **A** **G**
π : **N** **Y**

Emission: 0.25×0.4
Transition: 0.8×0.1 *0.008*

s : **A** **G**
π : **X** **Y**

0.0136

s : **A** **G**
π : **Y** **N**

Emission: 0.1×0.25
Transition: 0.2×0.2 *0.001*

s : **A** **G**
π : **N** **N**

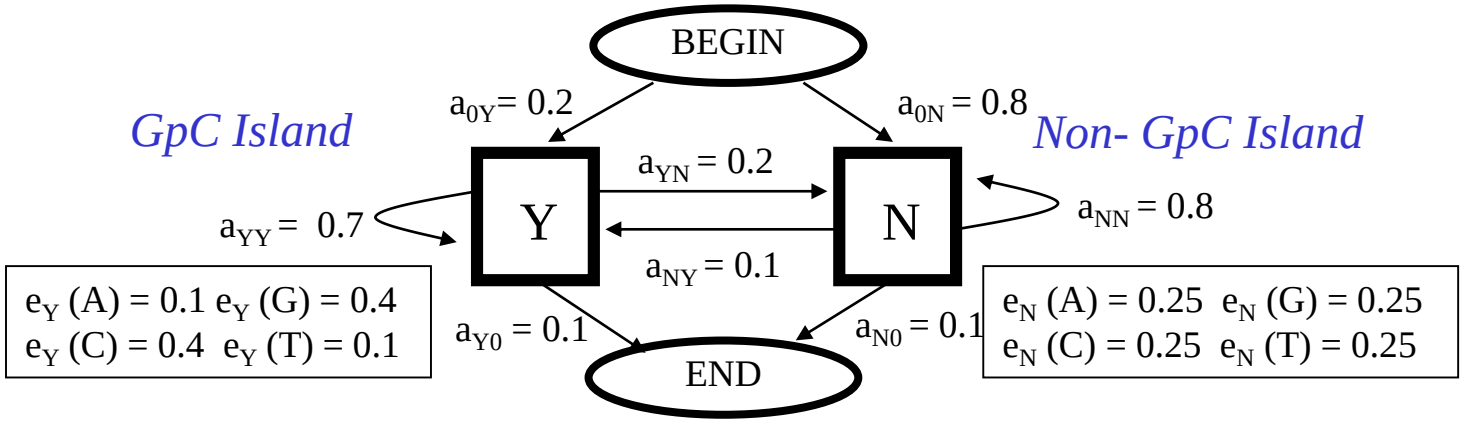
Emission: 0.25×0.25
Transition: 0.8×0.8 *0.04*

Sum

s : **A** **G**
π : **X** **N**

0.041

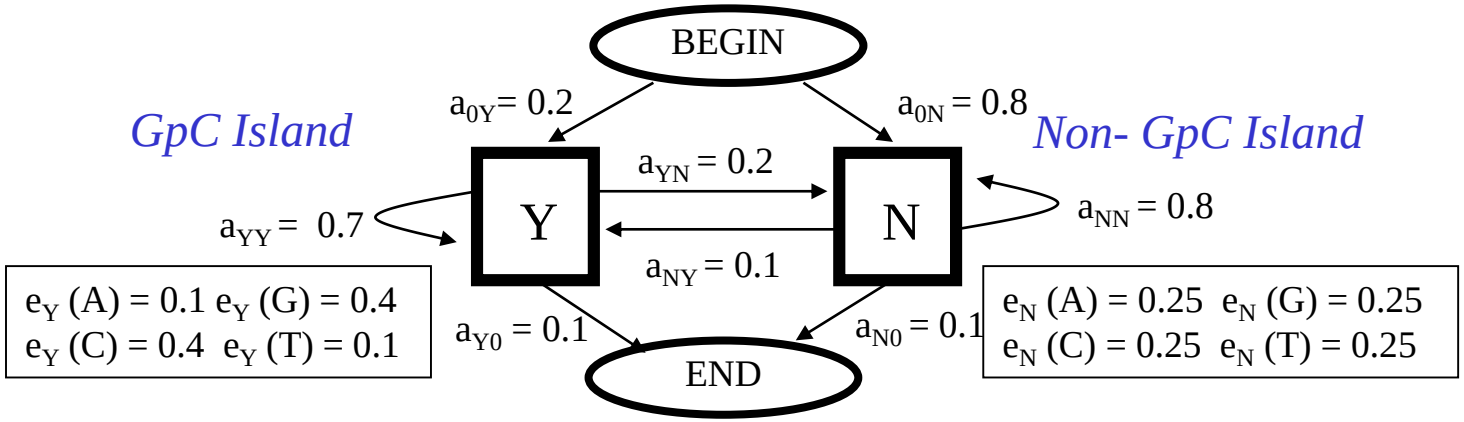
Strategy: Summing over all the possible paths



s :	**A**	**G**	**C**	s :	**A**	**G**	**C**
π :	**X**	**Y**	**Y**	π :	**X**	**N**	**Y**
	0.0136 ×		0.4	+	*0.041* ×		0.4
			0.7				0.1

s :	**A**	**G**	**C**	s :	**A**	**G**	**C**
π :	**X**	**Y**	**N**	π :	**X**	**N**	**N**
	0.0136 ×		0.25	+	*0.041* ×		0.25
			0.2				0.8

Strategy: Summing over all the possible paths

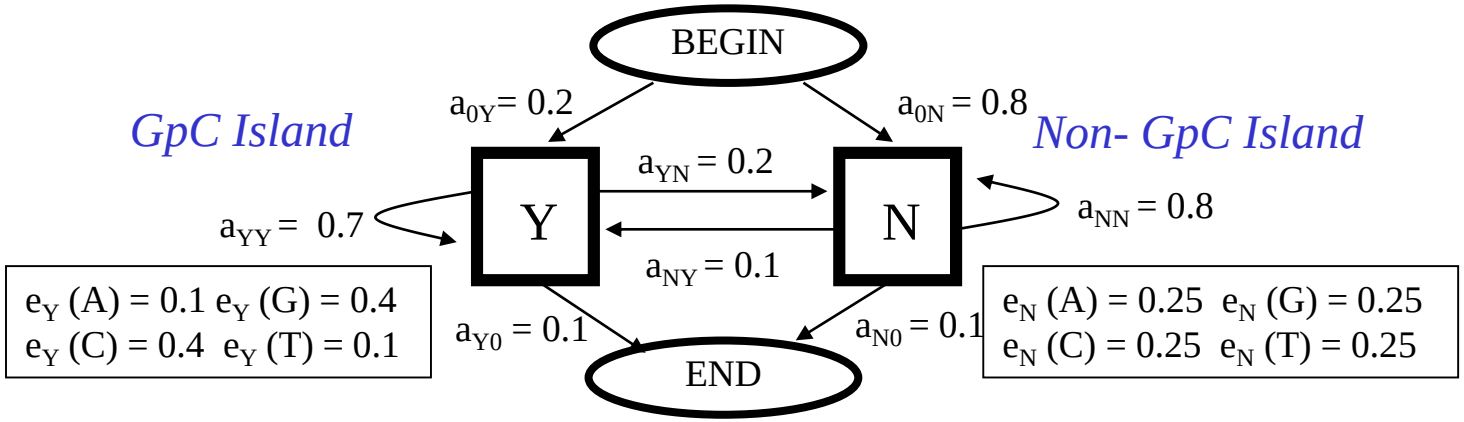


s :	A	G	C		s :	A	G	C		s :	A G C
π :	X	Y	Y		π :	X	N	Y		π :	X X Y
	0.0136 ×		0.4	+		0.041 ×		0.4			0.005448
			0.7					0.1			

Sum

s :	A	G	C		s :	A	G	C		s :	A G C
π :	X	Y	N		π :	X	N	N		π :	X X N
	0.0136 ×		0.25	+		0.041 ×		0.25			0.00888
			0.2					0.8			

Strategy: Summing over all the possible paths



Iterating until the last position of the sequence:

A	G	C	G	C	G	T	A	A	T	C	T	G	
X	X	X	X	X	X	X	X	X	X	X	X	Y	
												×	0.1 (a_{Y0})
													+
A	G	C	G	C	G	T	A	A	T	C	T	G	
X	X	X	X	X	X	X	X	X	X	X	X	N	
												×	0.1 (a_{N0})

P(s|M)

Forward Algorithm

On the basis of preceding observations the computation of P(s | M) can be decomposed in simplest problems

For each state k and each position i in the sequence, we compute:

$$F_k(i) = P(s^1 s^2 s^3 \ldots\ldots s^i, \pi(i) = k \mid M)$$

Will be understood

Initialisation: $F_{BEGIN}(0) = 1 \quad F_i(0) = 0 \quad \forall\ i \neq \text{BEGIN}$

Recurrence: $F_l(i+1) = P(s^1 s^2 \ldots s^i s^{i+1}, \pi(i+1) = l) =$

$$= \Sigma_k P(s^1 s^2 \ldots s^i, \pi(i) = k) \cdot a_{kl} \cdot e_l(s^{i+1}) =$$

$$= e_l(s^{i+1}) \cdot \Sigma_k F_k(i) \cdot a_{kl}$$

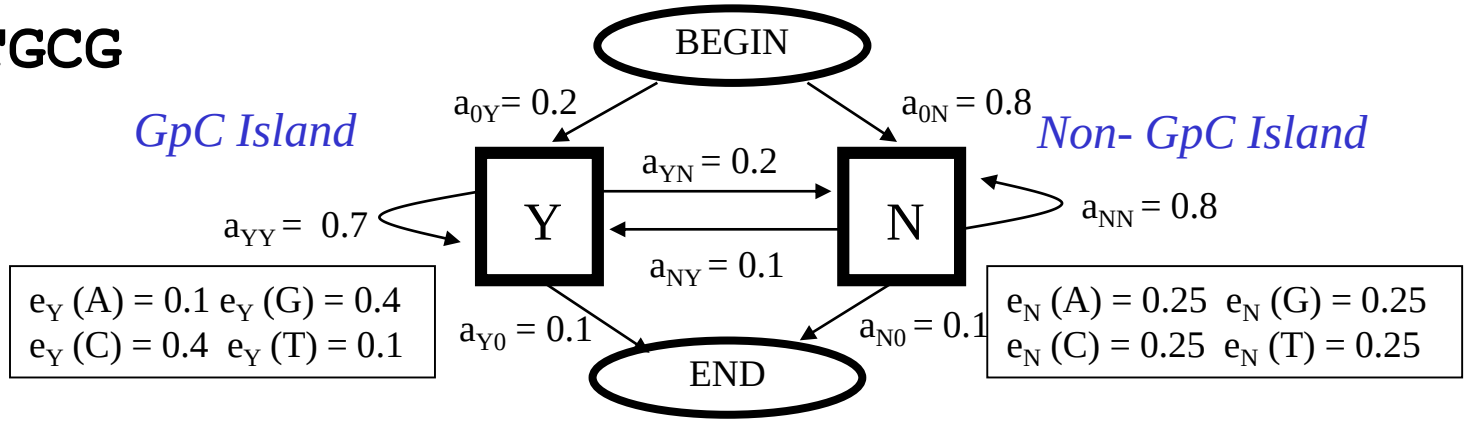
Termination: $P(s) = P(s^1 s^2 s^3 \ldots\ldots s^T, \pi(T+1) = END) =$

$$= \Sigma_k P(s^1 s^2 \ldots s^T, \pi(T) = k) \cdot a_{k0}$$

$$= \Sigma_k F_k(T) \cdot a_{k0}$$

Forward Algorithm: Example

S = ATGCG



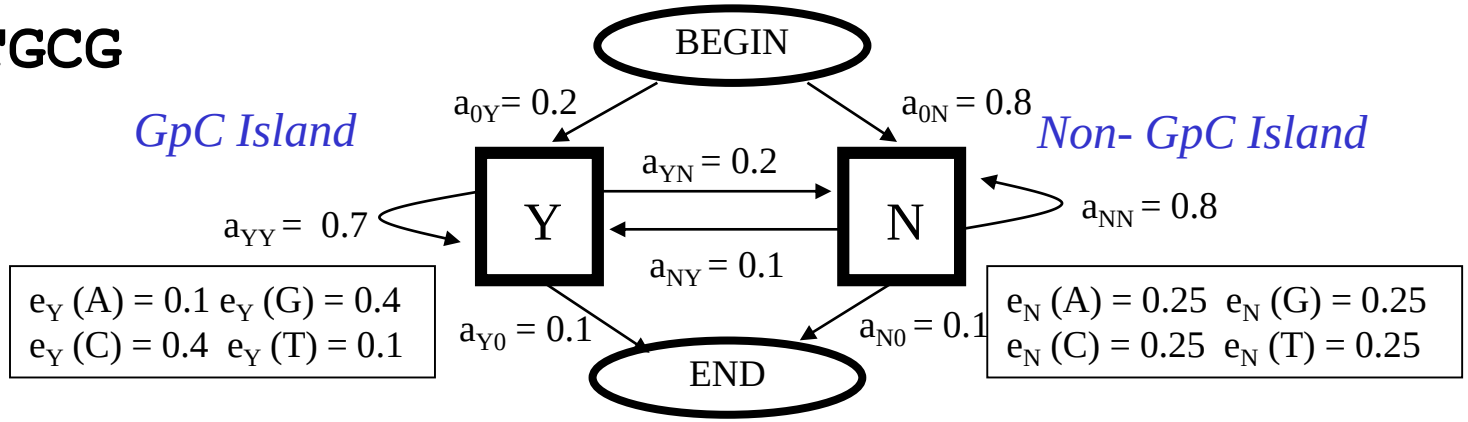
$$F_k(i) = \mathrm{P}(\, s^1 s^2 s^3 \ldots\ldots s^{\,i},\ \pi(i) = k)$$

Initialisation: $F_{BEGIN}(0) = 1 \quad F_i(0) = 0 \quad \forall\ i \neq \mathrm{BEGIN}$

	-	A	T	G	C	G	-
Begin	**1**						
Y	**0**						
N	**0**						
End	**0**						

Forward Algorithm: Example

S = ATGCG



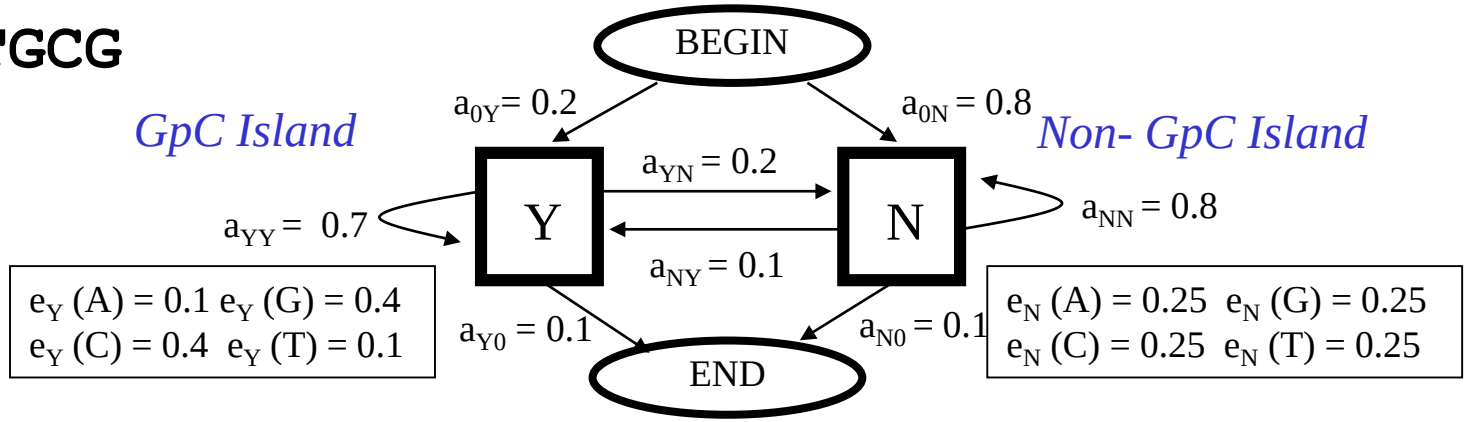
$$F_k(i) = P(s^1 s^2 s^3 \ldots\ldots s^i, \pi(i) = k)$$

Recurrence: $F_l(1) = e_l(s^1) \cdot \Sigma_k F_k(0) \cdot a_{kl}$

	-	A	T	G	C	G	-
Begin	**1**	**0**					
Y	**0**	**0.2x0.1= =2e-2**					
N	**0**	**0.8x0.25= =0.2**					
End	**0**	**0**					

Forward Algorithm: Example

S = ATGCG



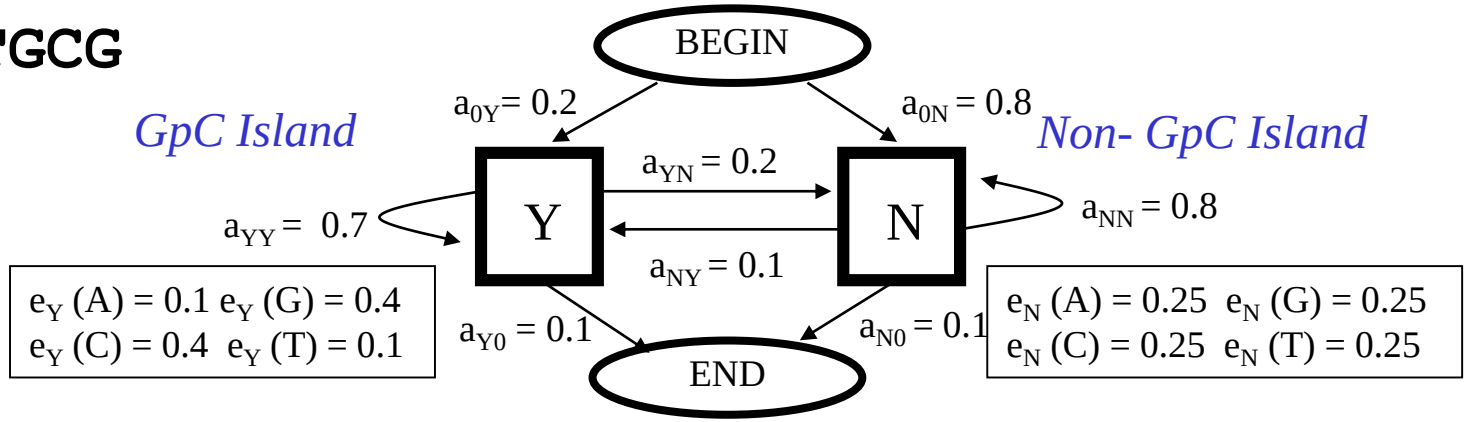
$$F_k(i) = P(s^1 s^2 s^3 \ldots\ldots s^i, \pi(i) = k)$$

Recurrence: $F_l(2) = e_l(s^2) \cdot \Sigma_k F_k(1) \cdot a_{kl}$

	-	A	T	G	C	G	-
Begin	**1**	**0**	**0**				
Y	**0**	**2e-2**	**2e-2x0.7x0.1+ +0.2x0.1x0.1= =3.4e-3**				
N	**0**	**0.2**	**2e-2x0.2x0.25+ +0.2x0.8x0.25= =4.1e-2**				
End	**0**	**0**	**0**				

Forward Algorithm: Example

S = ATGCG



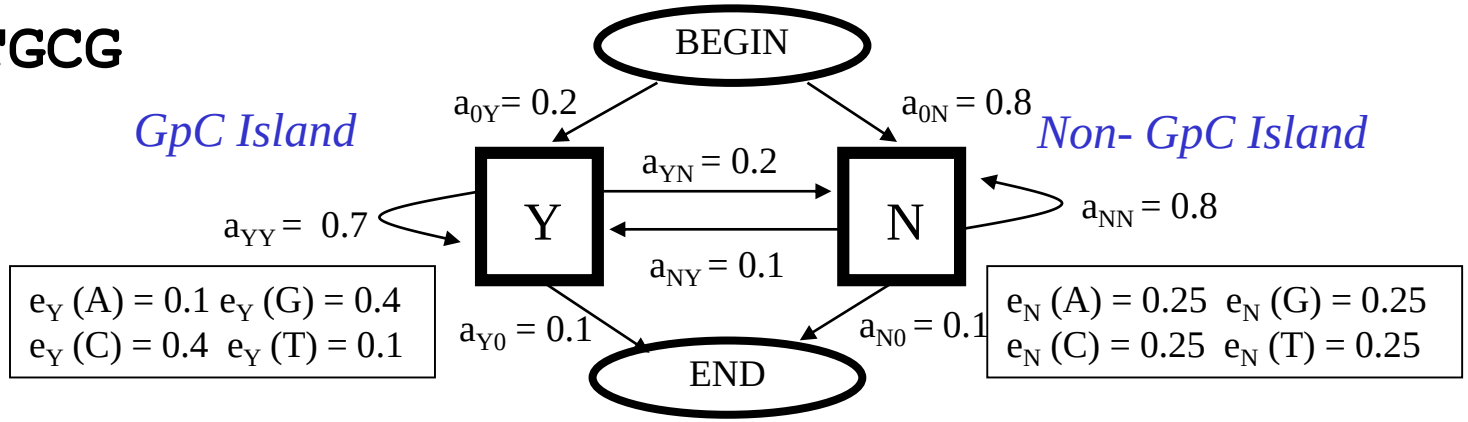
$$F_k(i) = P(s^1 s^2 s^3 \ldots\ldots s^i, \pi(i) = k)$$

Recurrence: $F_l(3) = e_l(s^3) \cdot \Sigma_k F_k(2) \cdot a_{kl}$

	-	A	T	G	C	G	-
Begin	**1**	**0**	**0**	**0**			
Y	**0**	**2e-2**	**3.4e-3**	**3.4e-3x0.7x0.4+ +4.1e-2x0.1x0.4= =2.592e-3**			
N	**0**	**0.2**	**4.1e-2**	**3.4e-3x0.2x0.25+ +4.1e-2x0.8x0.25= =8.37e-3**			
End	**0**	**0**	**0**	**0**			

Forward Algorithm: Example

S = ATGCG



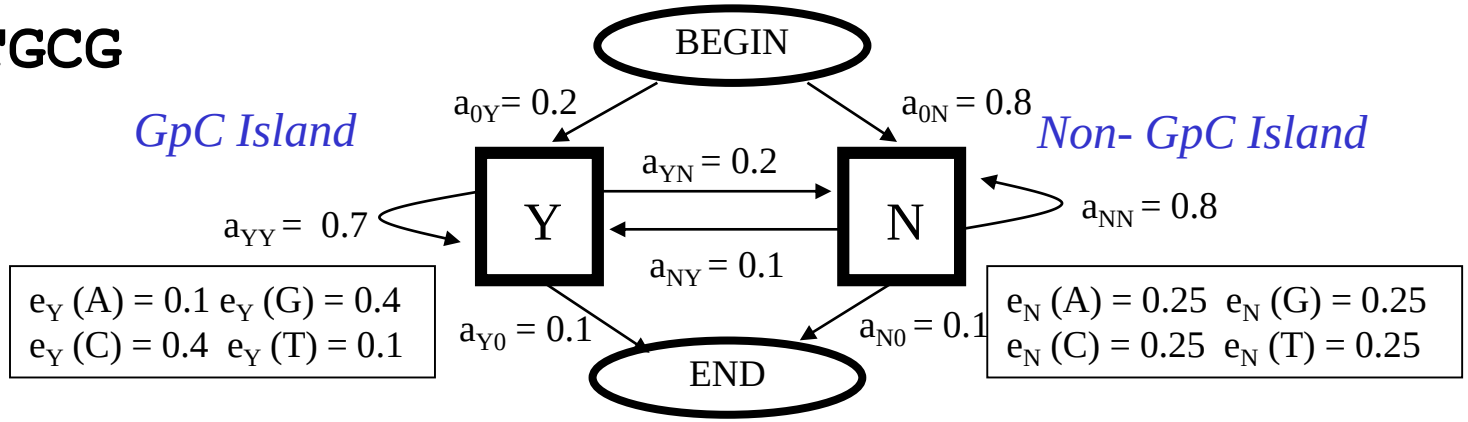
$$F_k(i) = P(s^1 s^2 s^3 \ldots\ldots s^i, \pi(i) = k)$$

Recurrence: $F_l(4) = e_l(s^4) \cdot \Sigma_k F_k(3) \cdot a_{kl}$

	-	A	T	G	C	G	-
Begin	**1**	**0**	**0**	**0**	**0**		
Y	**0**	**2e-2**	**3.4e-3**	**2.592e-3**	**2.592e-3x0.7x0.4+ +8.37e-3x0.1x0.4= =1.06056e-3**		
N	**0**	**0.2**	**4.1e-2**	**8.37e-3**	**2.592e-3x0.2x0.25+ +8.37e-3x0.8x0.25= =1.8036e-3**		
End	**0**	**0**	**0**	**0**	**0**		

Forward Algorithm: Example

S = ATGCG



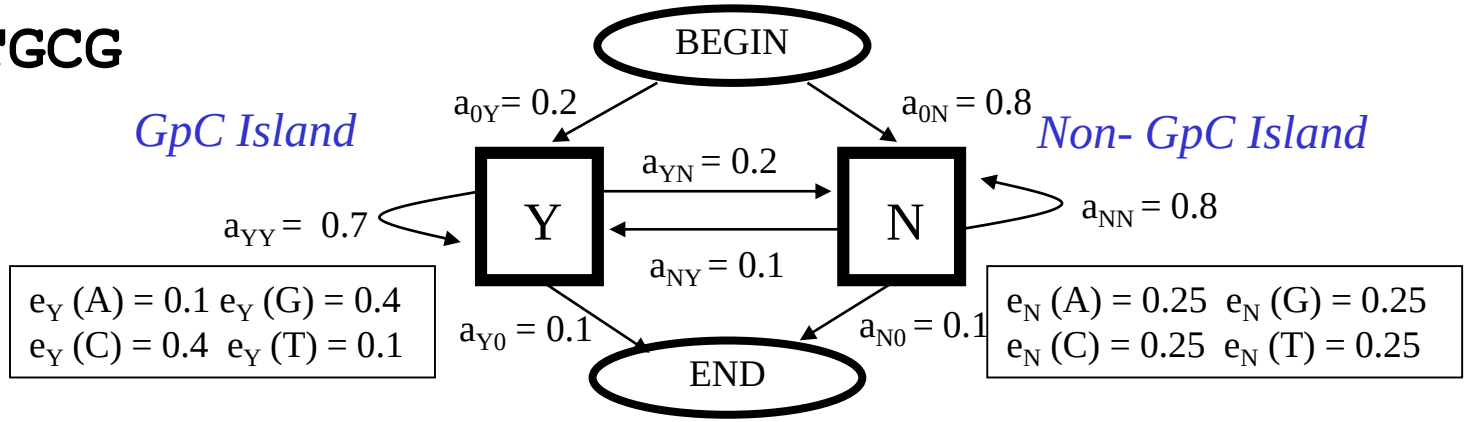
$$F_k(i) = P(s^1 s^2 s^3 \ldots\ldots s^i, \pi(i) = k)$$

Recurrence: $F_l(5) = e_l(s^5) \cdot \Sigma_k F_k(4) \cdot a_{kl}$

	-	A	T	G	C	G	-
Begin	**1**	**0**	**0**	**0**	**0**	**0**	
Y	**0**	**2e-2**	**3.4e-3**	**2.592e-3**	**1.06056e-3**	**1.06056e-3x0.7x0.4+ +1.8036e-3x0.1x0.4= =3.691008e-4**	
N	**0**	**0.2**	**4.1e-2**	**8.37e-3**	**1.8036e-3**	**1.06056e-3x0.2x0.25+ +1.8036e-3x0.8x0.25= =4.13748e-4**	
End	**0**	**0**	**0**	**0**	**0**	**0**	

Forward Algorithm: Example

S = ATGCG



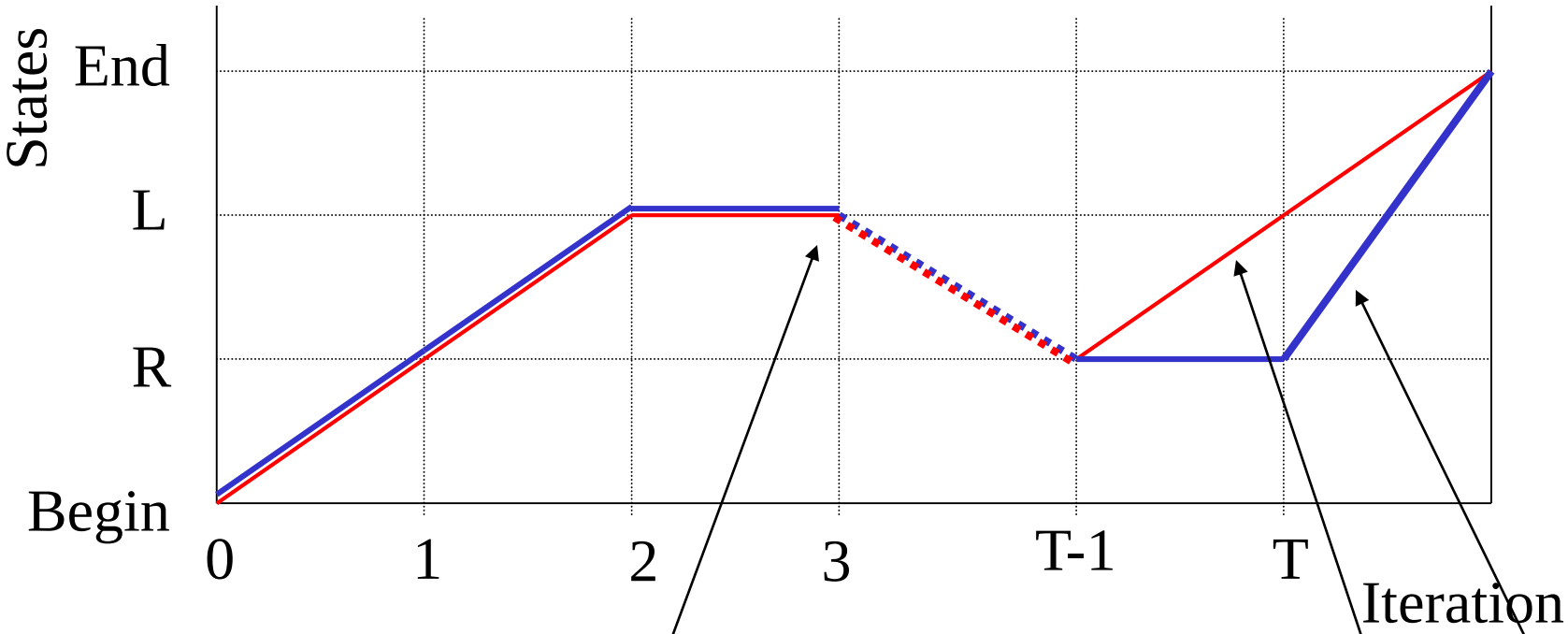
$$F_k(i) = P(s^1 s^2 s^3 \ldots\ldots s^i, \pi(i) = k)$$

Termination: $P(s) = \Sigma_k F_k(T) \cdot a_{k0}$

	-	A	T	G	C	G	-
Begin	**1**	**0**	**0**	**0**	**0**	**0**	**0**
Y	**0**	**2e-2**	**3.4e-3**	**2.592e-3**	**1.06056e-3**	**3.691008e-4**	**0**
N	**0**	**0.2**	**4.1e-2**	**8.37e-3**	**1.8036e-3**	**4.13748e-4**	**0**
End	**0**	**0**	**0**	**0**	**0**	**0**	**3.691008e-4x0.1+ +4.13748e-4x0.1= =7.828488e-5**

P(s|M)=7.828488e-5

Computing P(s,π | M) for each path is a redundant operation

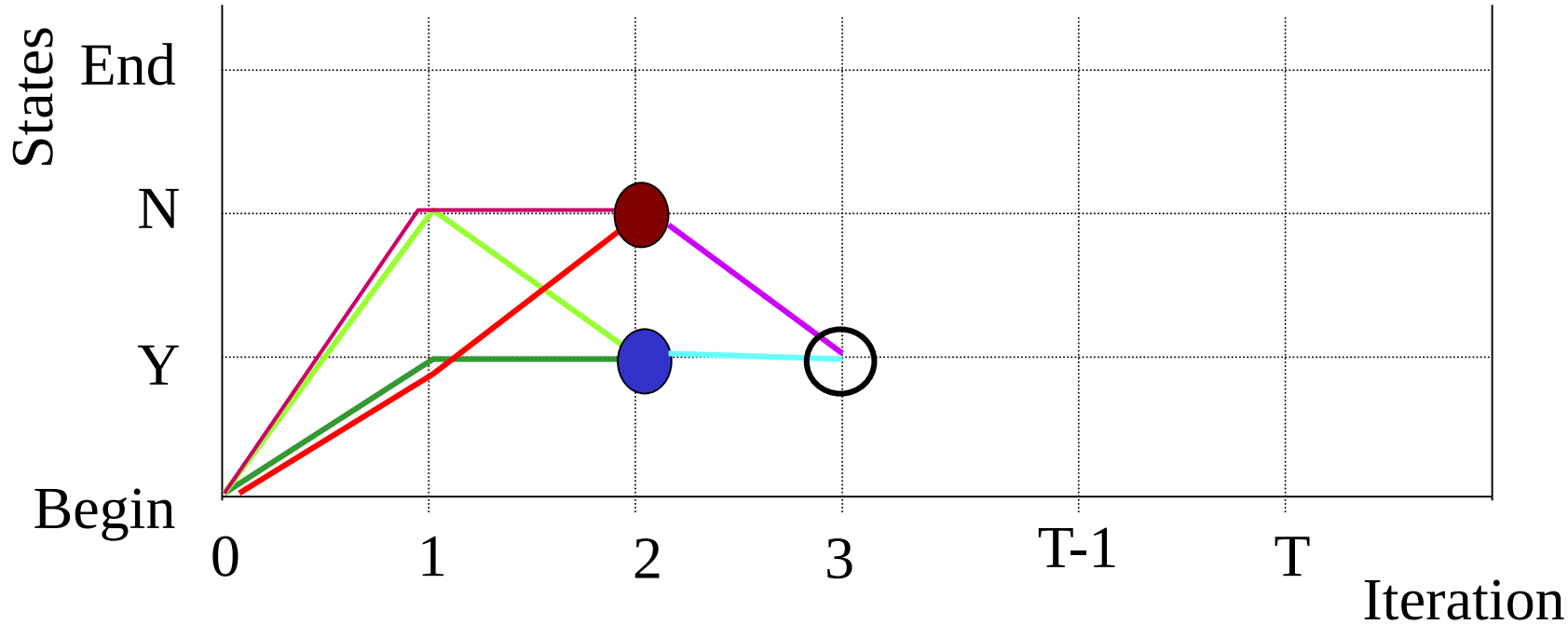


$$P(s,\pi_1 \mid M) = \left(\prod_{t=1}^{T-1} a_{\pi_1(t-1)\pi_1(t)} \cdot e_{\pi_1(t)}(s^t)\right) \cdot a_{\pi_1(T-1)\pi_1(T)} \cdot e_{\pi_1(T)}(s^T) \cdot a_{\pi_1(T)0}$$

$$P(s,\pi_2 \mid M) = \left(\prod_{t=1}^{T-1} a_{\pi_2(t-1)\pi_2(t)} \cdot e_{\pi_2(t)}(s^t)\right) \cdot a_{\pi_2(T-1)\pi_2(T)} \cdot e_{\pi_2(T)}(s^T) \cdot a_{\pi_2(T)0}$$

If we compute the common part only once we gain 2·(T-1) operations

Summing over all the possible paths



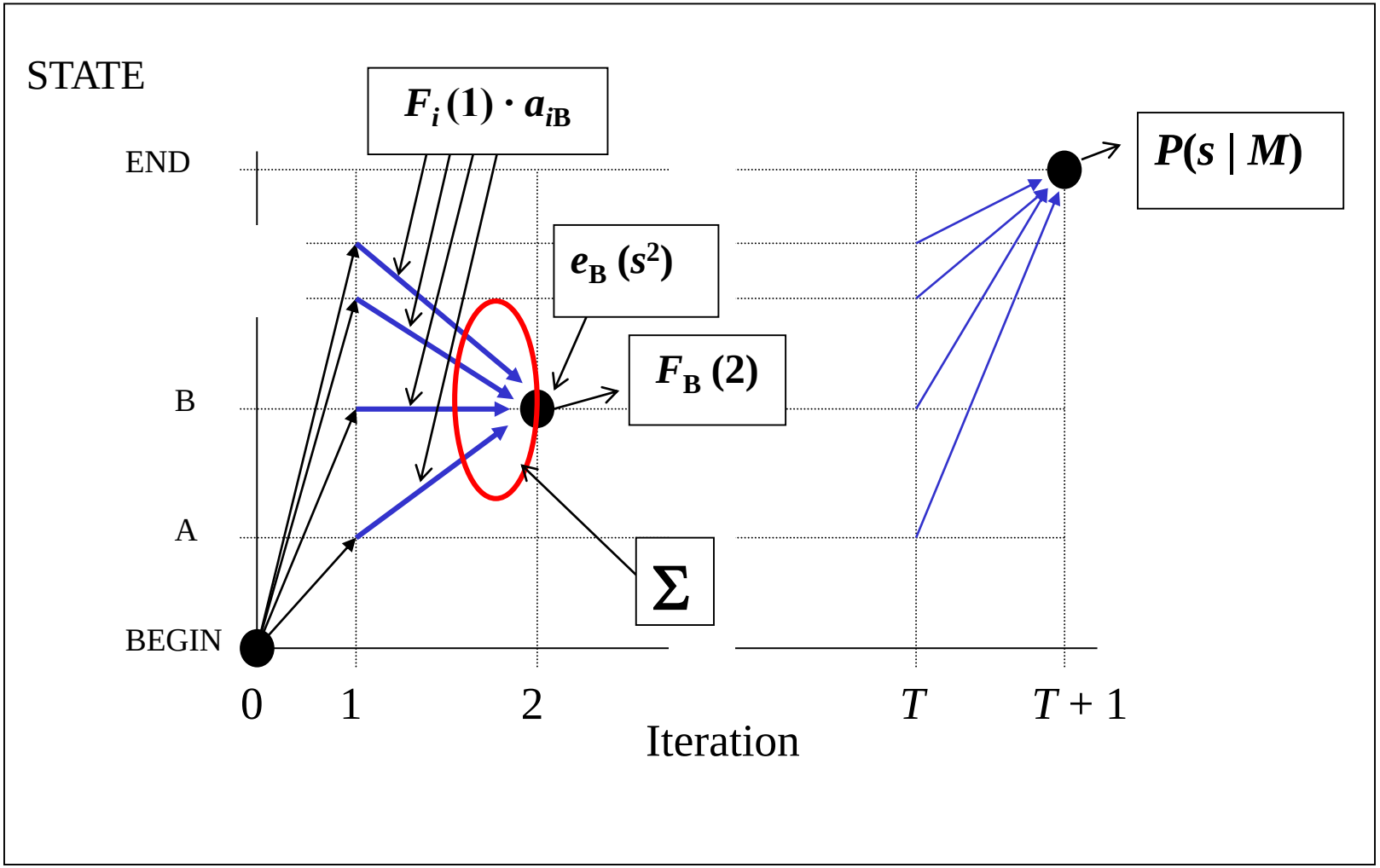
If we know the probabilities of emitting the two first characters of the sequence ending the path in states *L* and *R* respectively:

$$F_{\mathrm{Y}}(2) \equiv \mathrm{P}(s^1, s^2, \pi(2) = \mathrm{Y} \mid M) \quad \text{and } F_{\mathrm{N}}(2) \equiv \mathrm{P}(s^1, s^2, \pi(2) = \mathrm{N} \mid M)$$

then we can compute:

$$\mathrm{P}(s^1, s^2, s^3, \pi(3) = \mathrm{Y} \mid M) = F_{\mathrm{Y}}(2) \cdot a_{\mathrm{YY}} \cdot e_{\mathrm{Y}}(s^3) + F_{\mathrm{N}}(2) \cdot a_{\mathrm{NY}} \cdot e_{\mathrm{Y}}(s^3)$$

Forward Algorithm



Forward algorithm: computational complexity

Naïf method

$$P(s \mid M) = \Sigma_{\pi} P(s, \pi \mid M)$$

There are N^T possible paths.

Each path requires about $2 \cdot T$ operations.

The time for the computation is $O(T \cdot N^T)$

s :	**A**	**G**	**C**	**G**	**C**	**G**	**T**	**A**	**A**	**T**	**C**	**T**	**G**
π :	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**

Emission: $0.1 \times 0.4 \times 0.4 \times 0.4 \times 0.4 \times 0.4 \times 0.1 \times 0.1 \times 0.1 \times 0.1 \times 0.4 \times 0.1 \times 0.4$

Transition: $0.2 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.1$

s :	**A**	**G**	**C**	**G**	**C**	**G**	**T**	**A**	**A**	**T**	**C**	**T**	**G**
π :	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**N**

Emission: $0.1 \times 0.4 \times 0.4 \times 0.4 \times 0.4 \times 0.4 \times 0.1 \times 0.1 \times 0.1 \times 0.1 \times 0.4 \times 0.1 \times 0.25$

Transition: $0.2 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.7 \times 0.2 \times 0.1$

Forward algorithm: computational complexity

- *Forward algorithm*

T positions, ***N*** values for each position

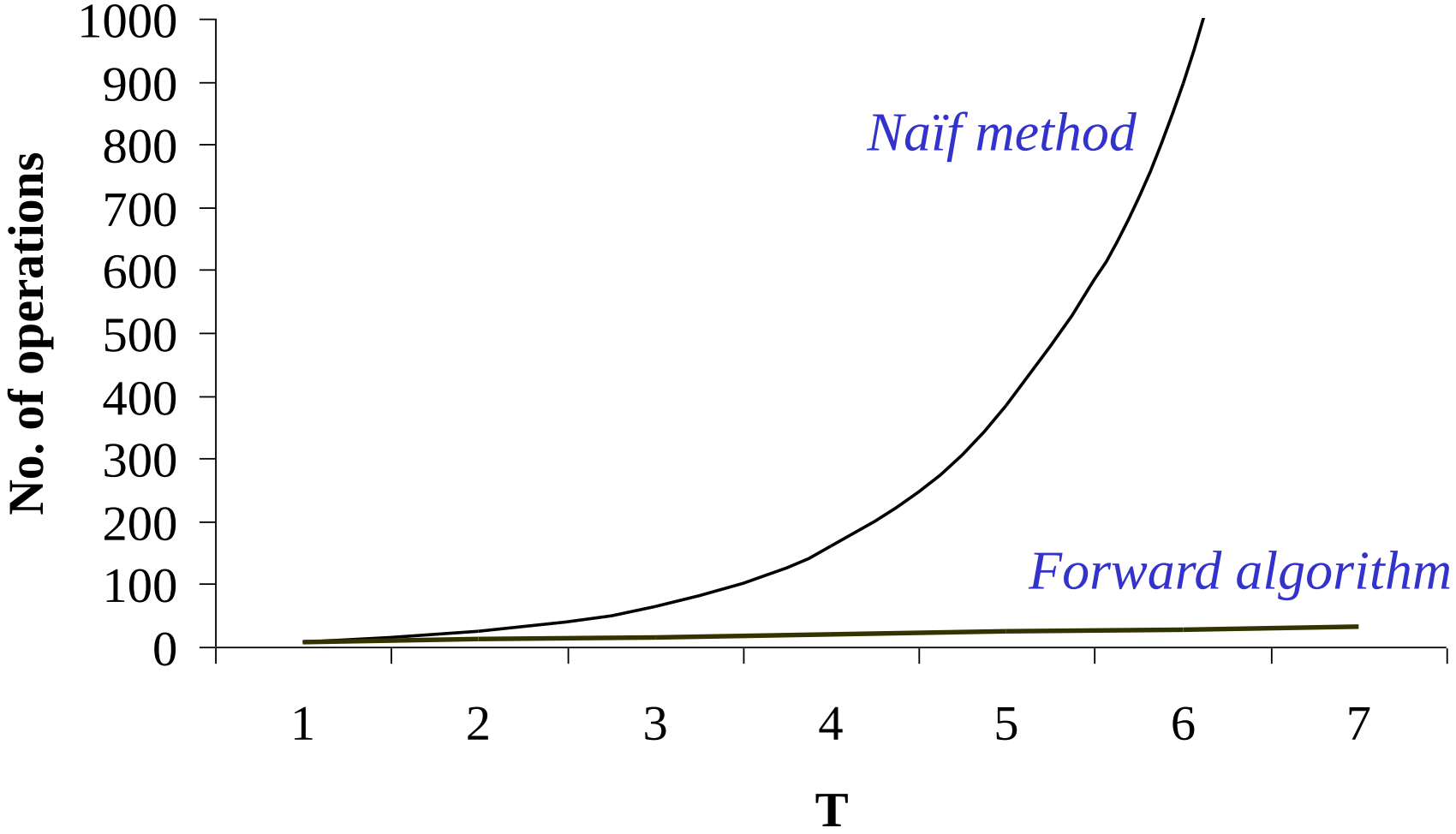
Each element requires about ***2·N*** product and **1** sum

The time for the computation is $O(T \cdot N^2)$

s :	A	G	C	s :	A	G	C	s :	A	G	C
π :	X	Y	Y	π :	X	N	Y	π :	X	X	Y
	0.0136 ×		0.4	+	0.041 ×		0.4			0.005448	
			0.7				0.1				
s :	A	G	C	s :	A	G	C	s :	A	G	C
π :	X	Y	N	π :	X	N	N	π :	X	X	N
	0.0136 ×		0.25	+	0.041 ×		0.25			0.00888	
			0.2				0.8				

Sum

Forward algorithm: computational complexity



Hidden Markov Models: Algorithms

- Resumé
- Evaluating $P(s \mid M)$: Forward Algorithm
- Evaluating $P(s \mid M)$: Backward Algorithm

Backward Algorithm

Similar to the Forward algorithm: it computes P(s | M), reconstructing the sequence from the end

For each state k and each position i in the sequence, we compute:

$$B_k(i) = P(s^{i+1}s^{i+2}s^{i+3}\ldots\ldots s^T \mid \pi(i) = k)$$

Initialisation: $B_k(T) = P(\pi(T+1) = \text{END} \mid \pi(T) = k) = a_{k0}$

Recurrence: $B_l(i\text{-}1) = P(s^i s^{i+1}\ldots s^T \mid \pi(i-1) = l) =$

$$= \Sigma_k P(s^{i+1}s^{i+2}\ldots s^T \mid \pi(i) = k) \cdot a_{lk} \cdot e_k(s^i) =$$

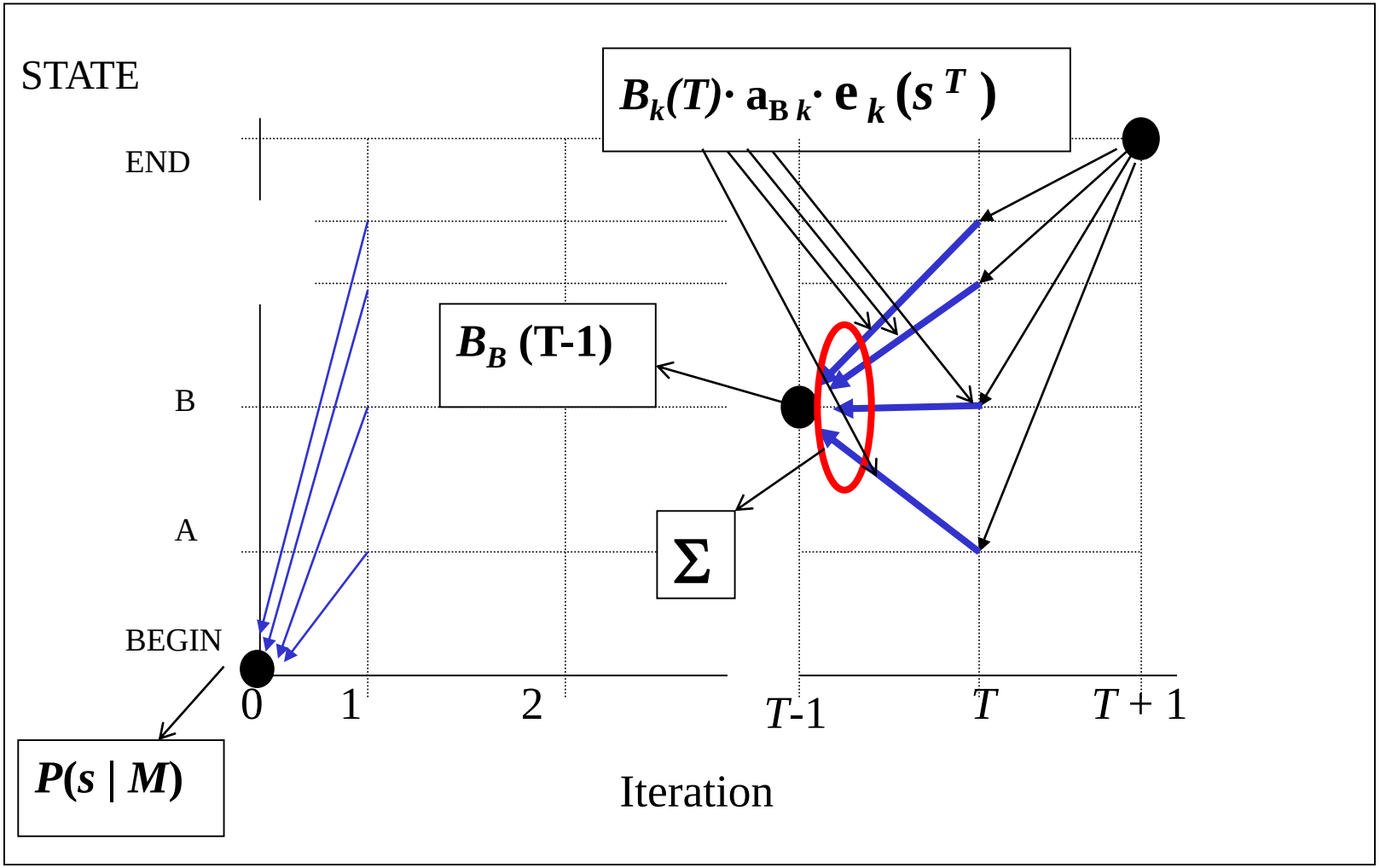
$$= \Sigma_k B_k(i) \cdot e_k(s^i) \cdot a_{lk}$$

Termination: $P(s) = P(s^1 s^2 s^3\ldots\ldots s^T \mid \pi(0) = BEGIN) =$

$$= \Sigma_k P(s^2\ldots s^T \mid \pi(1) = k) \cdot a_{0k} \cdot e_k(s^1) =$$

$$= \Sigma_k B_k(1) \cdot a_{0k} \cdot e_k(s^1)$$

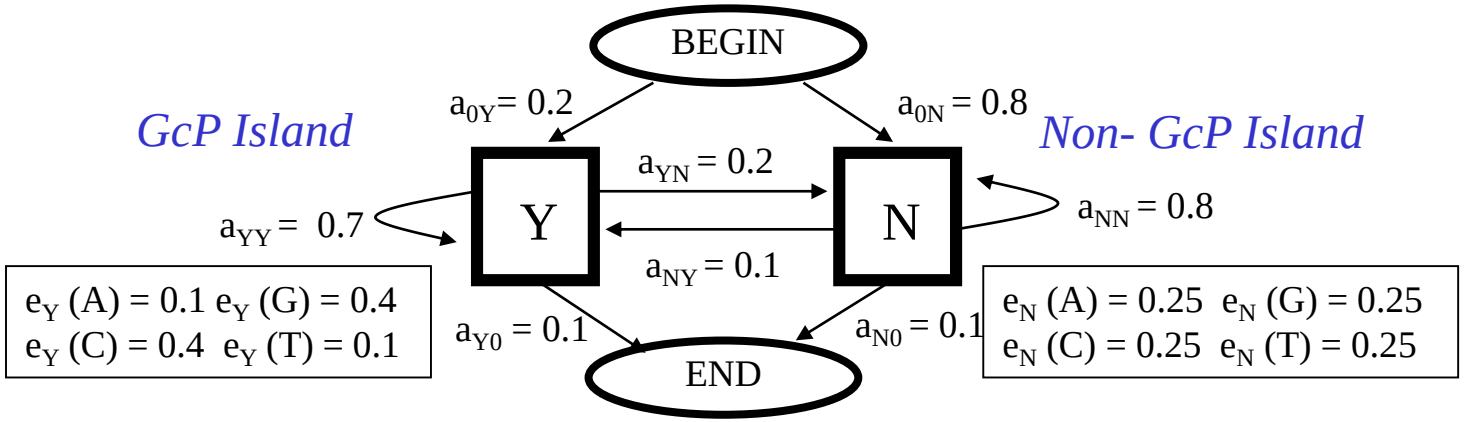
Backward Algorithm



Hidden Markov Models: Algorithms

- Resumé
- Evaluating $P(s \mid M)$: Forward Algorithm
- Evaluating $P(s \mid M)$: Backward Algorithm
- Showing the path: Viterbi decoding

Finding the best path



s : **A** **G**

π : **Y** **Y**

Emission: 0.1×0.4

Transition: 0.2×0.7 *0.0056*

s : **A** **G**

π : **N** **Y**

Emission: 0.25×0.4

Transition: 0.8×0.1 ***0.008***

s : **A** **G**

π : **Y** **N**

Emission: 0.1×0.25

Transition: 0.2×0.2 *0.001*

s : **A** **G**

π : **N** **N**

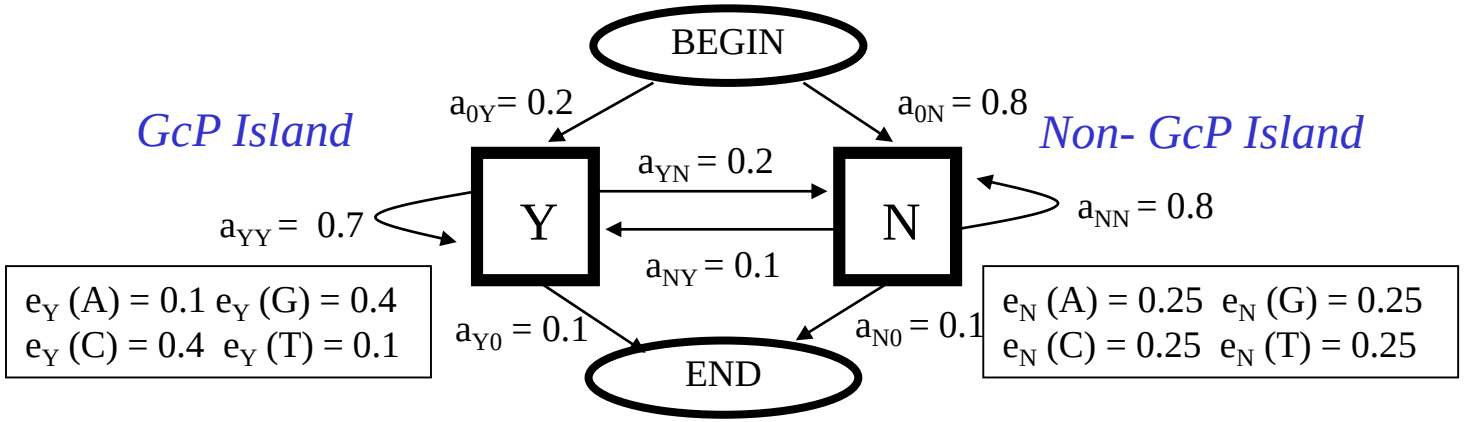
Emission: 0.25×0.25

Transition: 0.8×0.8 ***0.04***

In order to continue the process we only need to know the last state:

We can choose for each final state the path scoring with the maximum probability

Finding the best path



s : **A** **G**
π : **Y** **Y**

Emission: 0.1×0.4
Transition: 0.2×0.7 *0.0056*

s : **A** **G**
π : **N** **Y**

Emission: 0.25×0.4
Transition: 0.8×0.1 ***0.008***

s : **A** **G**
π : **N** **Y**

0.008

s : **A** **G**
π : **Y** **N**

Emission: 0.1×0.25
Transition: 0.2×0.2 *0.001*

s : **A** **G**
π : **N** **N**

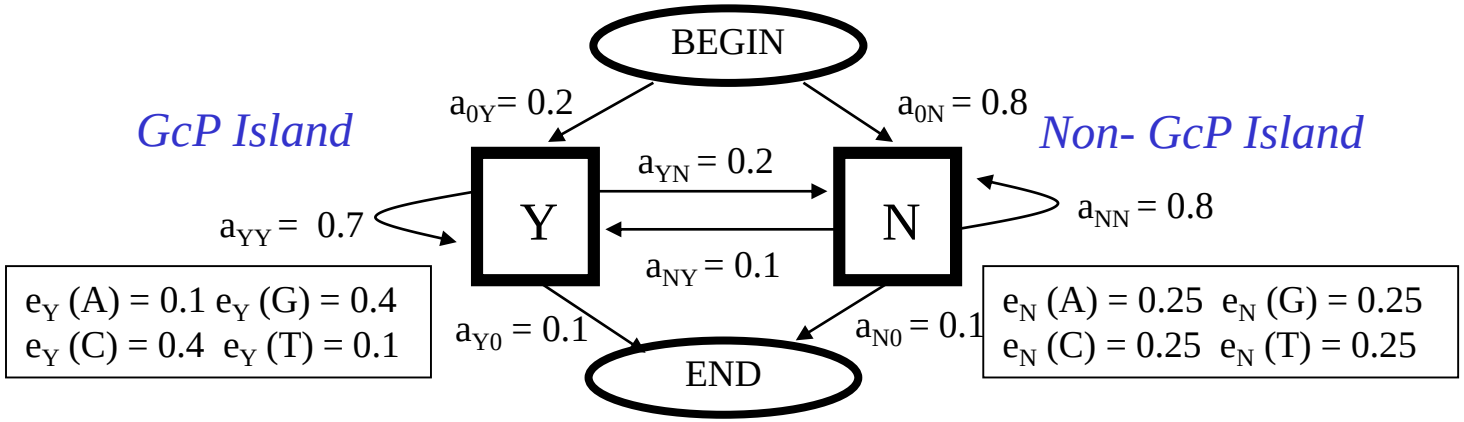
Emission: 0.25×0.25
Transition: 0.8×0.8 ***0.04***

Max

s : **A** **G**
π : **N** **N**

0.04

Finding the best path



s :	**A**	**G**	**C**
π :	**N**	**Y**	**Y**
	0.008 ×		0.4
			0.7
			= ***0.00224***

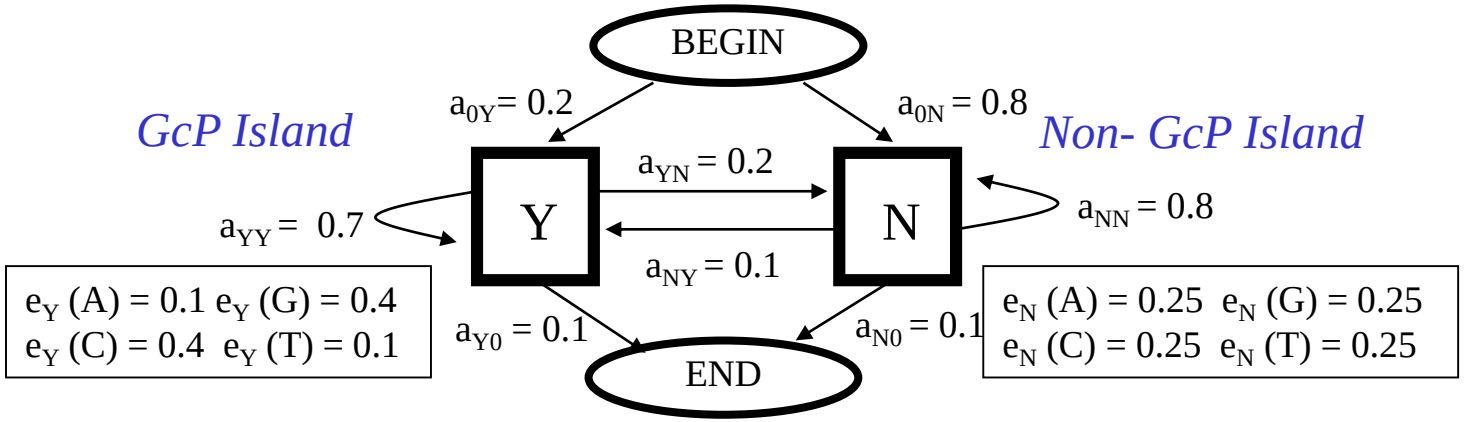
s :	**A**	**G**	**C**
π :	**N**	**N**	**Y**
	0.04 ×		0.4
			0.1
			= *0.0016*

s :	**A**	**G**	**C**
π :	**N**	**Y**	**N**
	0.008 ×		0.25
			0.2
			= *0.0004*

;

s :	**A**	**G**	**C**
π :	**N**	**N**	**N**
	0.04 ×		0.25
			0.8
			= ***0.008***

Finding the best path



s :	A	G	C
π :	N	Y	Y

$0.008 \times 0.4 \times 0.7 = \mathbf{0.00224}$

s :	A	G	C
π :	N	N	Y

$0.04 \times 0.4 \times 0.1 = 0.0016$

s :	A	G	C
π :	N	Y	N

$0.008 \times 0.25 \times 0.2 = 0.0004$;

s :	A	G	C
π :	N	N	N

$0.04 \times 0.25 \times 0.8 = \mathbf{0.008}$

Max

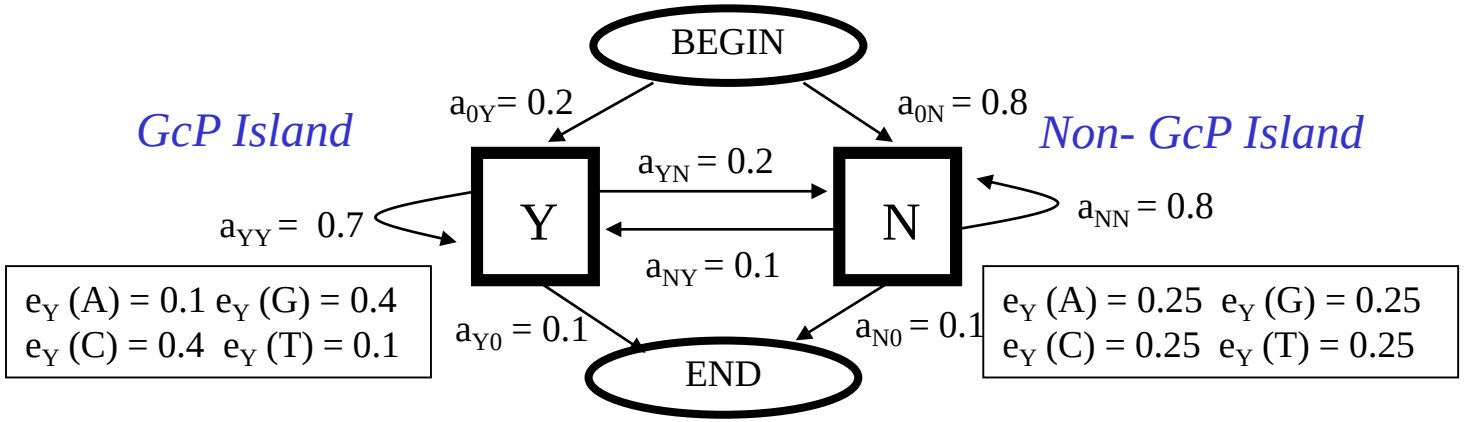
s :	A	G	C
π :	N	Y	Y

0.00224

s :	A	G	C
π :	N	N	N

0.008

Finding the best path



Iterating until the last position of the sequence:

A	G	C	G	C	G	T	A	A	T	C	T	G	
N	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	
												×	0.1 (a_{Y0})
A	G	C	G	C	G	T	A	A	T	C	T	G	
N	N	N	N	N	N	N	N	N	N	N	N	N	
												×	0.1 (a_{N0})

Choose the Maximum

Viterbi Algorithm

$\pi^* = \text{argmax}_{\pi} [\, P(\pi, s \mid M) \,]$

The computation of $P(s, \pi^* \mid M)$ can be decomposed in simplest problems

Let $V_k(i)$ be the probability of the most probable path for generating the subsequence $s^1 s^2 s^3 \ldots\ldots s^i$ ending in the state k at iteration i

Initialisation: $V_{BEGIN}(0) = 1 \quad V_i(0) = 0 \quad \forall\ i \neq \text{BEGIN}$

Recurrence: $V_l(i+1) = e_l(s^{i+1}) \cdot \text{Max}_k(V_k(i) \cdot a_{kl})$

$\text{ptr}_i(l) = \text{argmax}_k(V_k(i) \cdot a_{kl})$

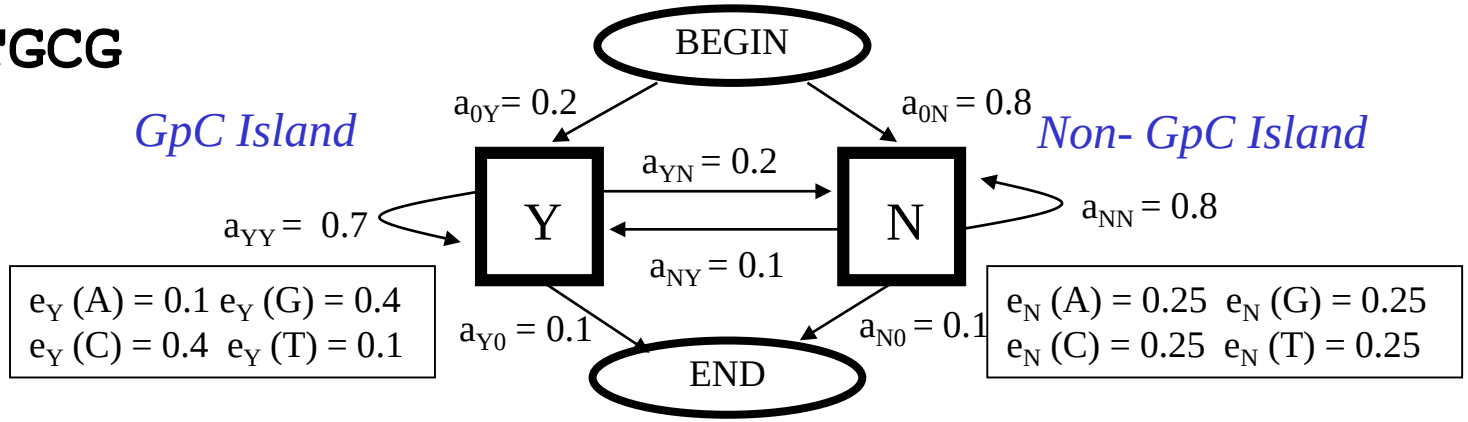
Termination: $P(s, \pi^*) = \text{Max}_k(V_k(T) \cdot a_{k0})$

$\pi^*(T) = \text{argmax}_k(V_k(T) \cdot a_{k0})$

Traceback: $\pi^*(i\text{-}1) = \text{ptr}_i(\pi^*(i))$

Viterbi Algorithm: Example

S = ATGCG

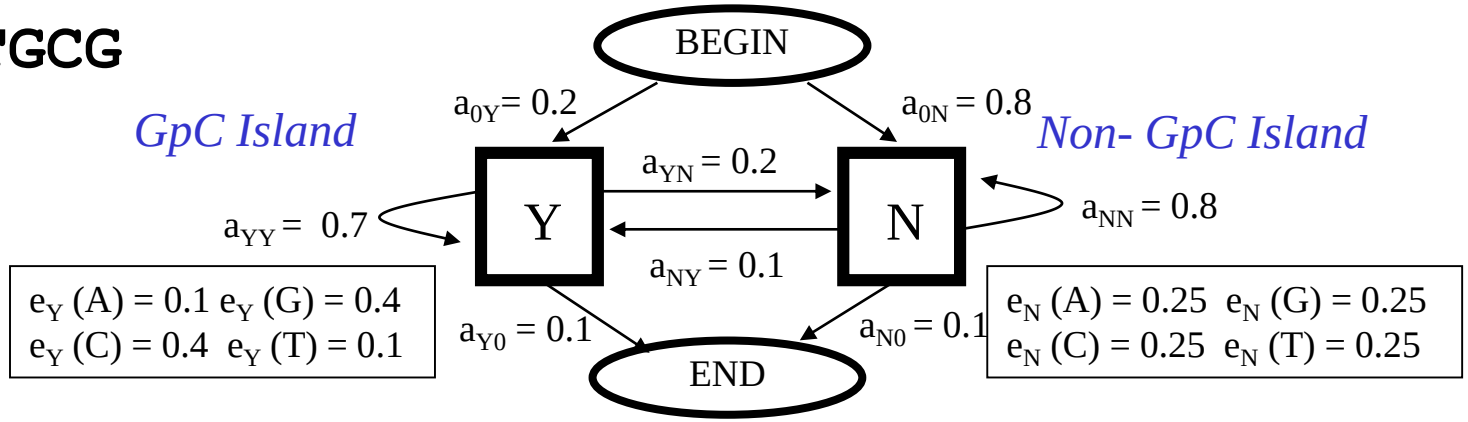


Initialisation: $V_{BEGIN}(0) = 1 \quad V_i(0) = 0 \quad \forall\ i \neq \text{BEGIN}$

	-	A	T	G	C	G	-
Begin	**1**						
Y	**0**						
N	**0**						
End	**0**						

Viterbi Algorithm: Example

S = ATGCG



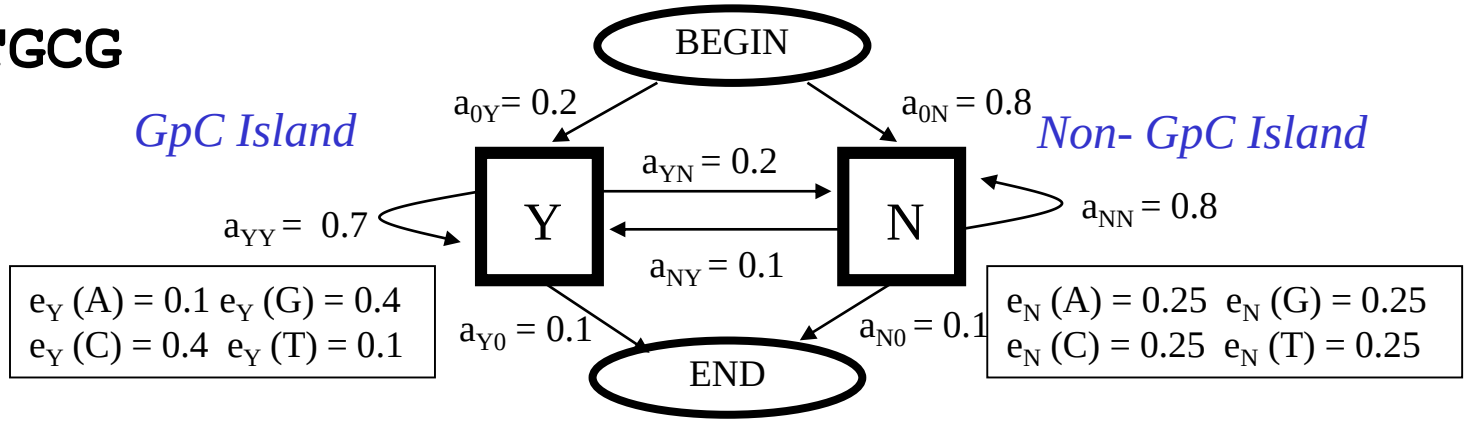
Recurrence: $V_l(1) = e_l(s^1) \cdot \mathrm{Max}_k(V_k(0) \cdot a_{kl})$

$$\mathrm{ptr}_1(l) = \mathrm{argmax}_k(V_k(0) \cdot a_{kl})$$

	-	A	T	G	C	G	-
Begin	**1**	**0**					
Y	**0**	**0.2x0.1= =2e-2 ptr=Begin**					
N	**0**	**0.8x0.25= =0.2 ptr=Begin**					
End	**0**	**0**					

Viterbi Algorithm: Example

S = ATGCG



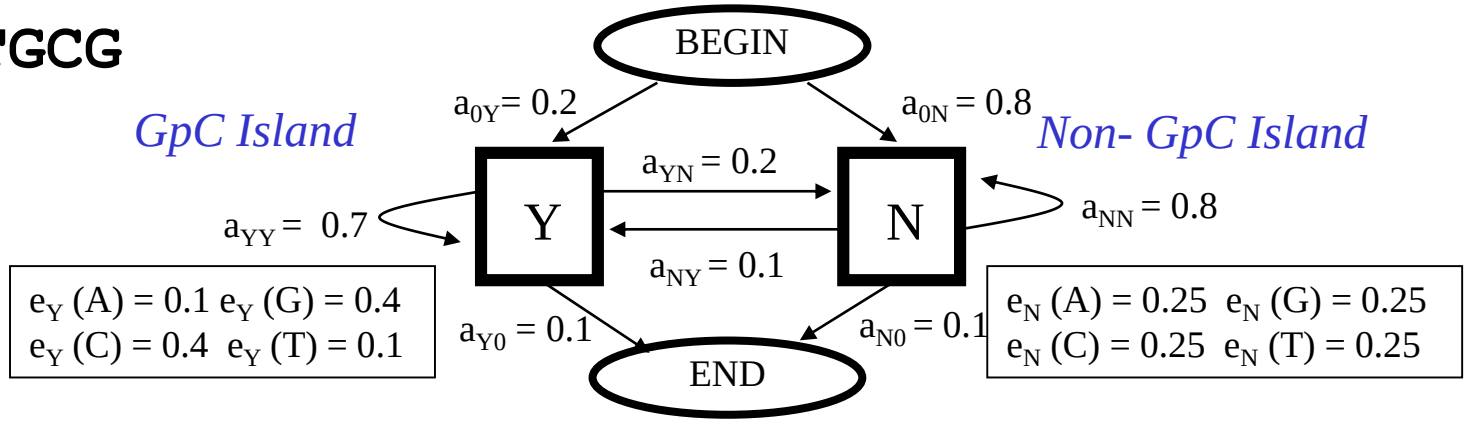
Recurrence: $V_l(2) = e_l(s^2) \cdot \mathrm{Max}_k(V_k(1) \cdot a_{kl})$

$$\mathrm{ptr}_2(l) = \mathrm{argmax}_k(V_k(1) \cdot a_{kl})$$

	-	A	T	G	C	G	-
Begin	**1**	**0**	**0**				
Y	**0**	**2e-2 Begin**	**Max(2e-2x0.7x0.1; 0.2x0.1x0.1) 2e-3; ptr=N**				
N	**0**	**0.2 Begin**	**Max(2e-2x0.2x0.25; 0.2x0.8x0.25) 1.6e-2; ptr=N**				
End	**0**	**0**	**0**				

Viterbi Algorithm: Example

S = ATGCG



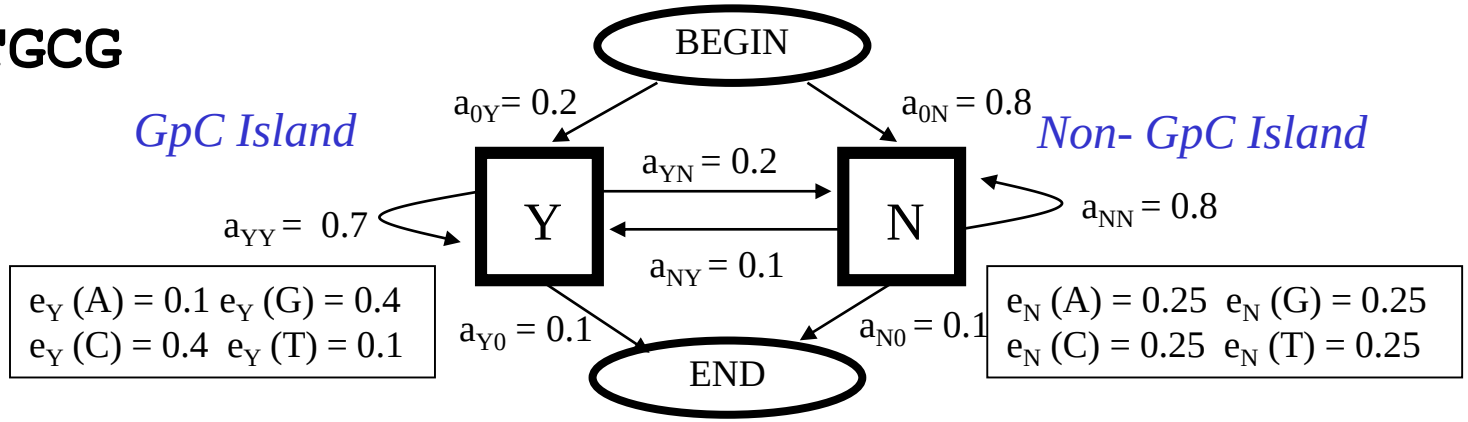
Recurrence: $V_l(3) = e_l(s^3) \cdot \mathrm{Max}_k(V_k(2) \cdot a_{kl})$

$$\mathrm{ptr}_3(l) = \mathrm{argmax}_k(V_k(2) \cdot a_{kl})$$

	-	A	T	G	C	G	-
Begin	**1**	**0**	**0**	**0**			
Y	**0**	**2e-2 Begin**	**2e-3; ptr=N**	**Max(2e-3x0.7x0.4; 1.6e-2x0.1x0.4) 6.4e-4; ptr=N**			
N	**0**	**0.2 Begin**	**1.6e-2; ptr=N**	**Max(2e-3x0.2x0.25; 1.6e-2x0.8x0.25) 3.2e-4; ptr=N**			
End	**0**	**0**	**0**	**0**			

Viterbi Algorithm: Example

S = ATGCG



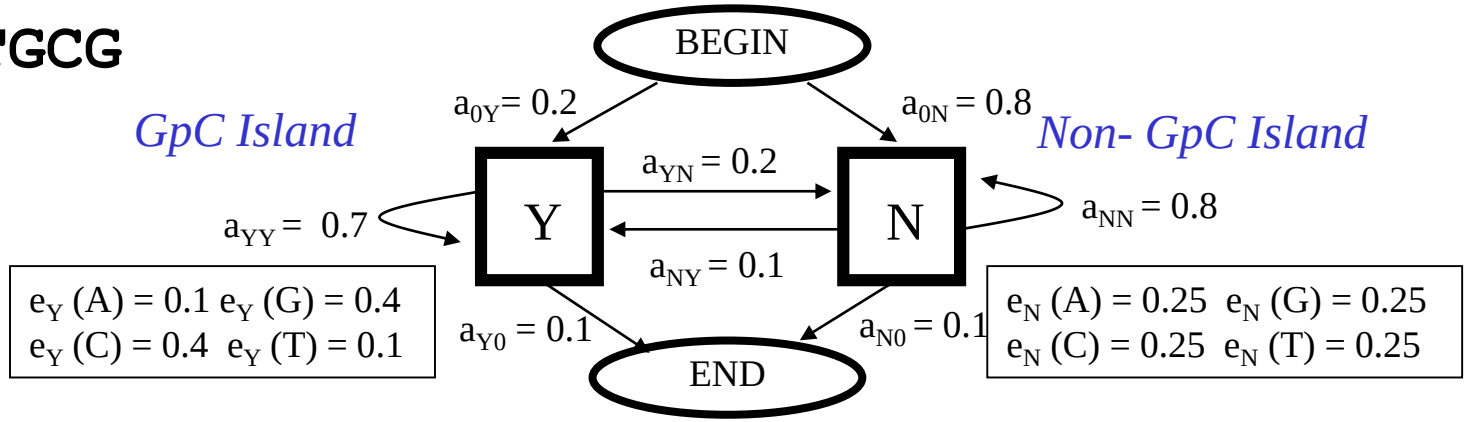
Recurrence: $V_l(4) = e_l(s^4) \cdot \mathrm{Max}_k(V_k(3) \cdot a_{kl})$

$$\mathrm{ptr}_4(l) = \mathrm{argmax}_k(V_k(3) \cdot a_{kl})$$

	-	A	T	G	C	G	-
Begin	**1**	**0**	**0**	**0**	**0**		
Y	**0**	**2e-2 Begin**	**2e-3; ptr=N**	**6.4e-4; ptr=N**	**Max(6.4e-4x0.7x0.4; 3.2e-4x0.1x0.4) 1.792e-4; ptr=Y**		
N	**0**	**0.2 Begin**	**1.6e-2; ptr=N**	**3.2e-4; ptr=N**	**Max(6.4e-4x0.2x0.25; 3.2e-4x0.8x0.25) 6.4e-5; ptr=N**		
End	**0**	**0**	**0**	**0**	**0**		

Viterbi Algorithm: Example

S = ATGCG



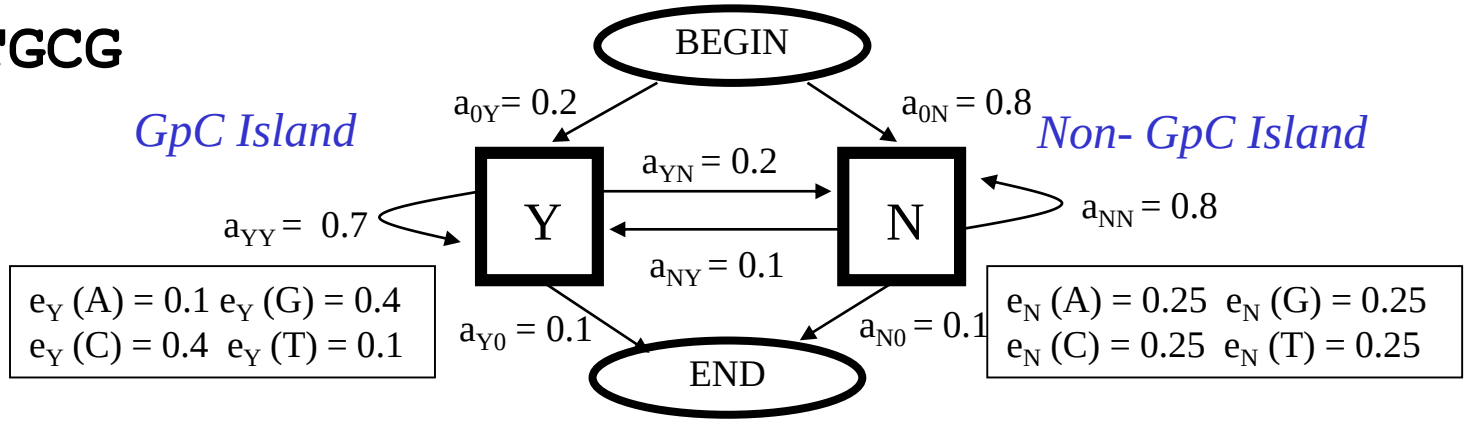
Recurrence: $V_l(5) = e_l(s^5) \cdot \mathrm{Max}_k(V_k(4) \cdot a_{kl})$

$$\mathrm{ptr}_5(l) = \mathrm{argmax}_k(V_k(4) \cdot a_{kl})$$

	-	A	T	G	C	G	-
Begin	**1**	**0**	**0**	**0**	**0**	**0**	
Y	**0**	**2e-2 Begin**	**2e-3; ptr=N**	**6.4e-4; ptr=N**	**1.792e-4; ptr=Y**	**Max(1.792e-4x0.7x0.4; 6.4e-5x0.1x0.4) 5.0176e-5; ptr=Y**	
N	**0**	**0.2 Begin**	**1.6e-2; ptr=N**	**3.2e-4; ptr=N**	**6.4e-5; ptr=N**	**Max(1.792e-4x0.2x0.25; 6.4e-5x0.8x0.25) 1.28e-5; ptr=N**	
End	**0**	**0**	**0**	**0**	**0**	**0**	

Viterbi Algorithm: Example

S = ATGCG



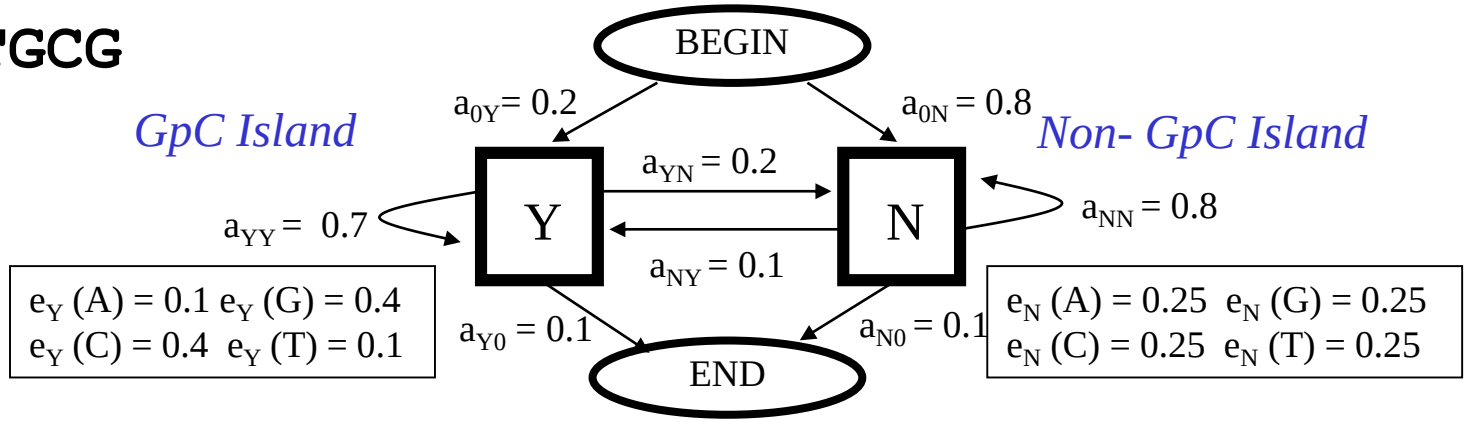
Termination: $P(s, \pi^*) = \mathrm{Max}_k (V_k(T) \cdot a_{k0})$

$$\pi^*(T) = \mathrm{argmax}_k (V_k(T) \cdot a_{k0})$$

	-	A	T	G	C	G	-
Begin	**1**	**0**	**0**	**0**	**0**	**0**	**0**
Y	**0**	**2e-2 Begin**	**2e-3; ptr=N**	**6.4e-4; ptr=N**	**1.792e-4; ptr=Y**	**5.0176e-5; ptr=Y**	**0**
N	**0**	**0.2 Begin**	**1.6e-2; ptr=N**	**3.2e-4; ptr=N**	**6.4e-5; ptr=N**	**1.28e-5; ptr=N**	**0**
End	**0**	**0**	**0**	**0**	**0**	**0**	**Max(5.0176e-5x0.1; 1.28e-5x0.1) 5.0176e-6; ptr=Y**

Viterbi Algorithm: Example

S = ATGCG



Traceback: $\pi^*(i\text{-}1) = \text{ptr}_i(\pi^*(i))$

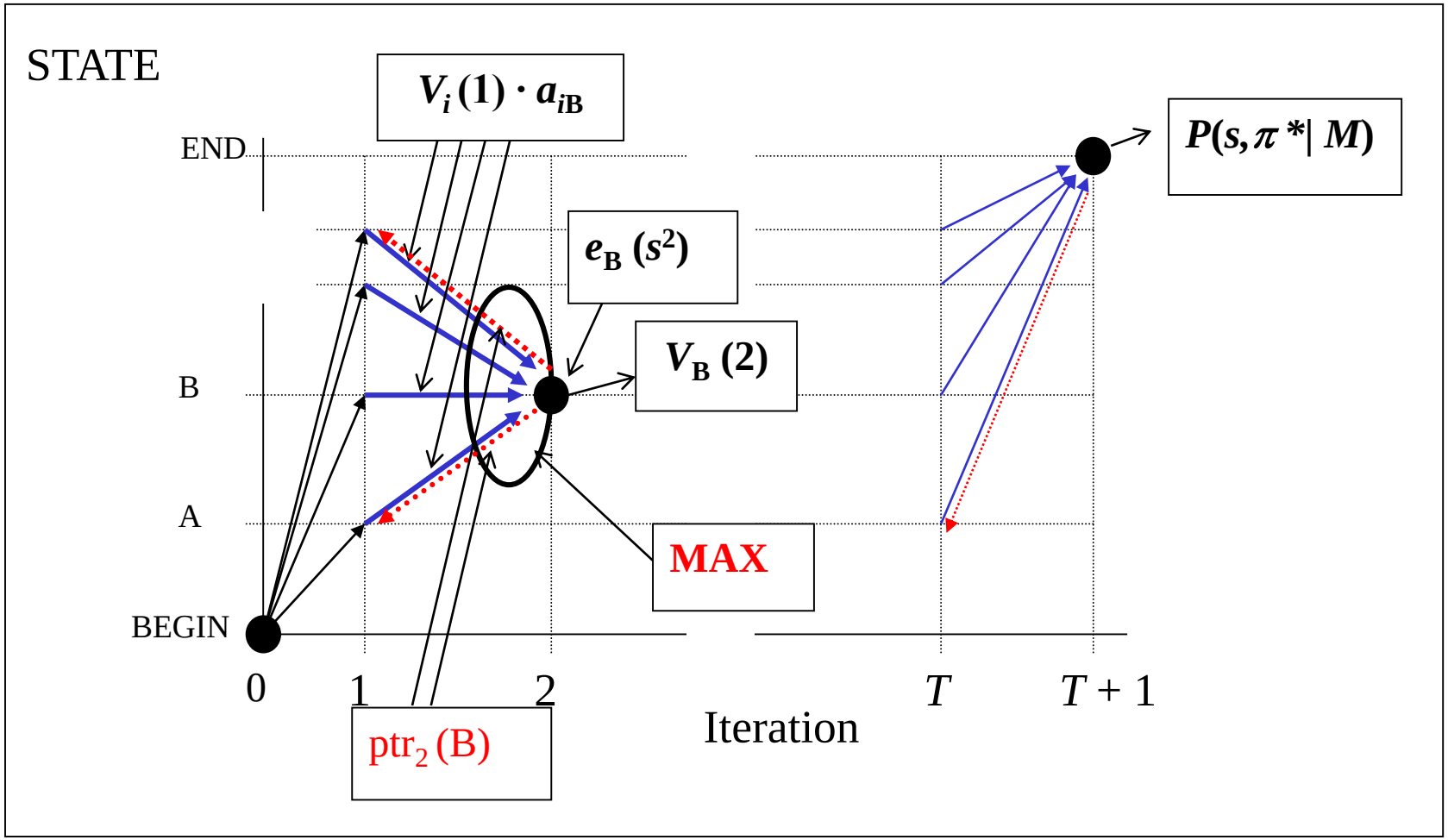
	-	A	T	G	C	G	-
Begin	**1**	**0**	**0**	**0**	**0**	**0**	**0**
Y	**0**	**2e-2 Begin**	**2e-3; ptr=N**	**6.4e-4; ptr=N**	**1.792e-4; ptr=Y**	**5.0176e-5; ptr=Y**	**0**
N	**0**	**0.2 Begin**	**1.6e-2; ptr=N**	**3.2e-4; ptr=N**	**6.4e-5; ptr=N**	**1.28e-5; ptr=N**	**0**
End	**0**	**0**	**0**	**0**	**0**	**0**	**5.0176e-6; ptr=Y**

S = - A T G C G -

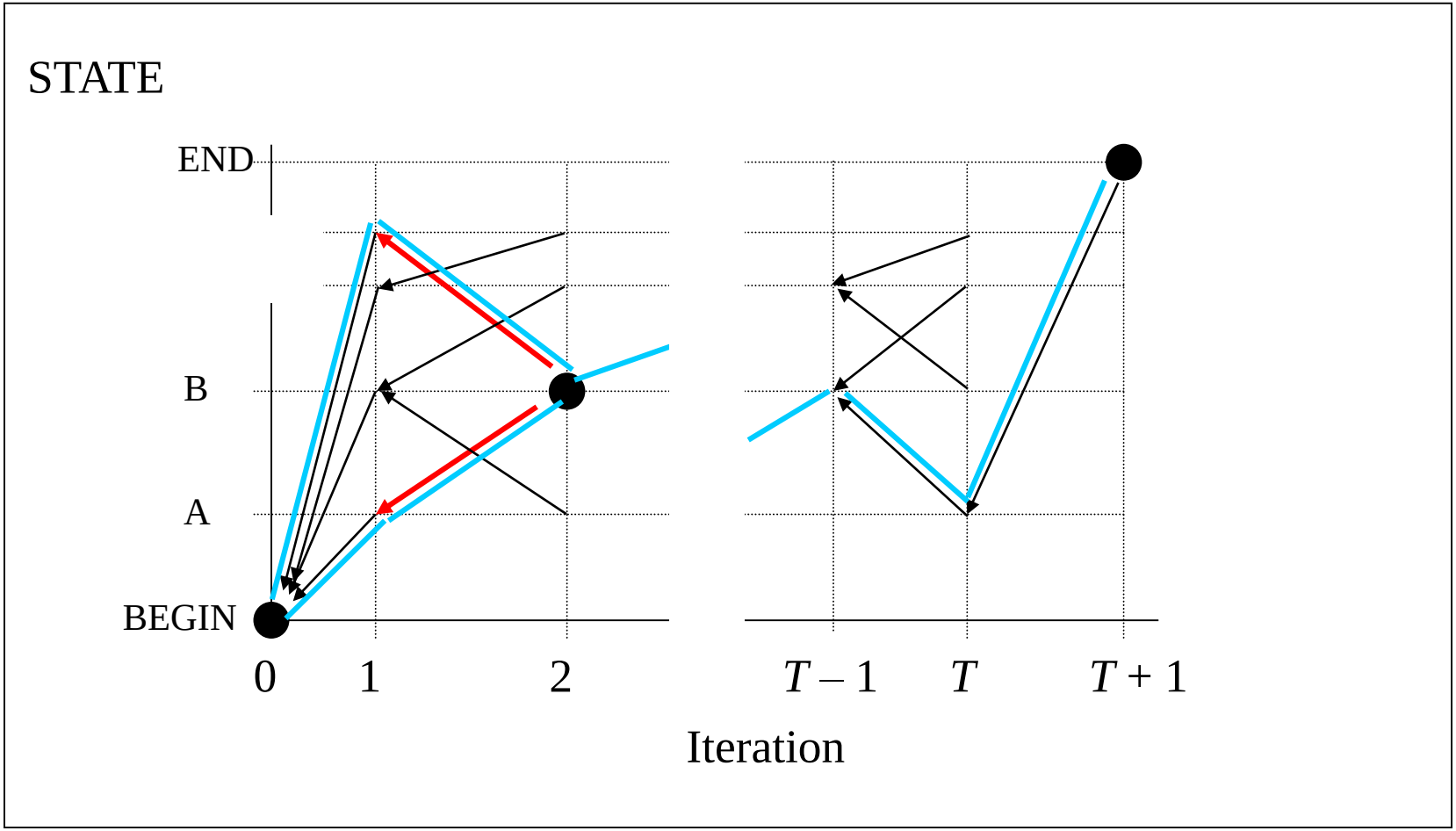
π* = B N N Y Y Y E

$P(s,\pi^*|M)=5.0176e\text{-}6$

Viterbi Algorithm



Viterbi Algorithm



Viterbi path

Different paths can have the same probability

An alternative: A posteriori decoding

For each position choose the state $\underline{\pi}(t)$:

$$\underline{\pi}(i) = \text{argmax}_k [\ \text{P}(\ \pi(i) = k | s, M\ )\ ]$$

How to compute $\text{P}(\ \pi(i) = k | s, M\ )$ for any state k and any position i?

$$P(\pi(i) = k \mid s, M) = \frac{P(\pi(i) = k, s \mid M)}{P(s \mid M)}$$

$$P(\pi(i) = k, s \mid M) = P(s^1 s^2 \ldots s^i, \pi(i) = k \mid M) \cdot P(s^{i+1}, s^{1+2}, \ldots s^T \mid \pi(i) = k, M) =$$

$$= F_k(i) \cdot B_k(i)$$

$$P(\pi(i) = k \mid s, M) = \frac{F_k(i) \cdot B_k(i)}{P(s \mid M)}$$

Elements of the Forward and Backward matrices

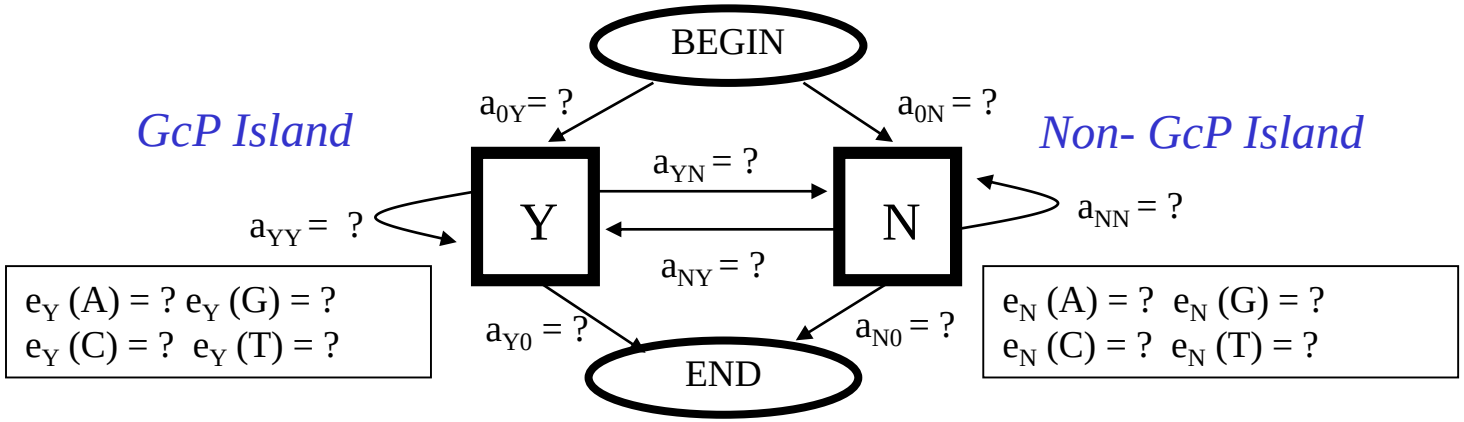
Computed with Forward or Backward algorithm termination steps

NB: this decoding can contain "non allowed" transitions

Hidden Markov Models: Algorithms

- Resumé
- Evaluating $P(s \mid M)$: Forward Algorithm
- Evaluating $P(s \mid M)$: Backward Algorithm
- Showing the path: Viterbi decoding
- Showing the path: A posteriori decoding
- Training a model: EM algorithm

If we know the path generating the training sequence



s :	**A**	**G**	**C**	**G**	**C**	**G**	**T**	**A**	**A**	**T**	**C**	**T**	**G**
π :	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**N**	**N**	**N**	**N**	**N**	**N**

Emission: $e_Y(A) \times e_Y(G) \times e_Y(A) \times e_Y(G) \times e_Y(C) \times e_Y(G) \times e_Y(T) \times e_N(A) \times e_N(A) \times e_N(T) \times e_N(C) \times e_N(T) \times e_N(G)$

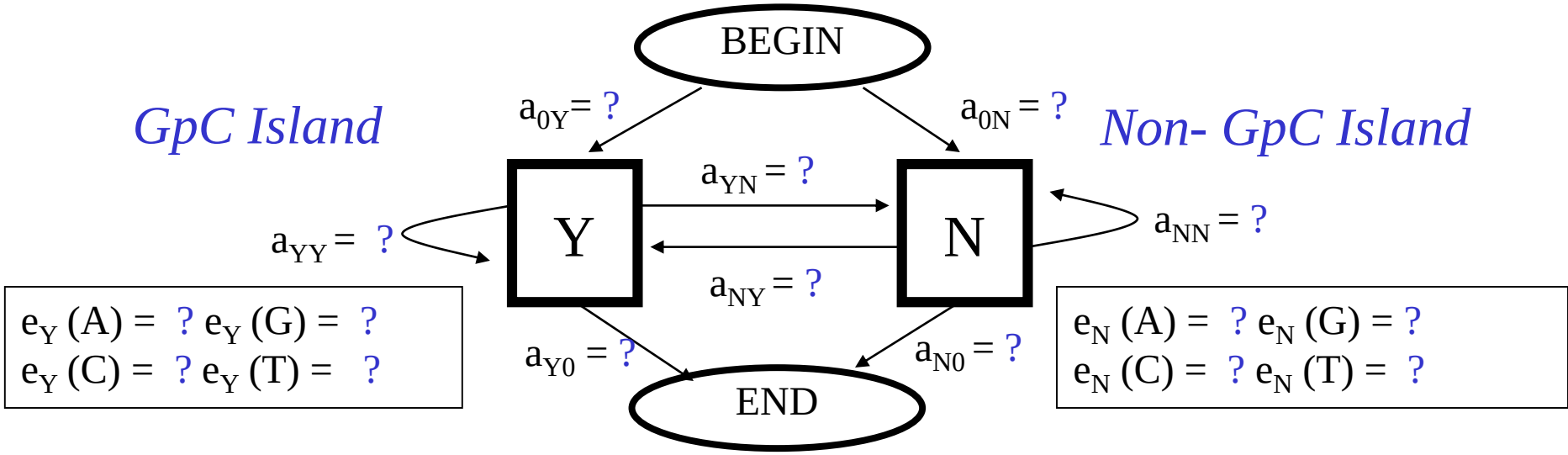
Transition: $a_{0Y} \times a_{YY} \times a_{YY} \times a_{YY} \times a_{YY} \times a_{YY} \times a_{YY} \times a_{YN} \times a_{NN} \times a_{NN} \times a_{NN} \times a_{NN} \times a_{NN} \times a_{N0}$

Just count!

Example: $a_{YY} = n_{YY} / (n_{YY} + n_{YN}) = 6/7$

$e_Y(A) = n_Y(A) / [n_Y(A) + n_Y(C) + n_Y(G) + n_Y(T)] = 1/7$

GpC Island, simple model

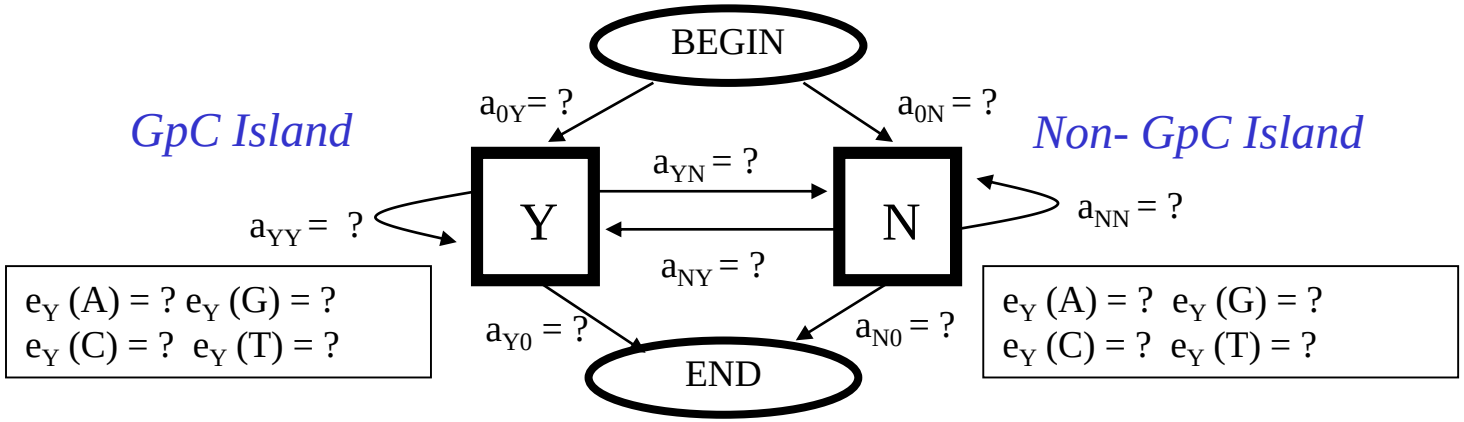


s :**AGCGCGTAATCTG**

π :**YYYYYYYNNNNNN**

- ***P(s, π | M) can be easily computed***
- ***How to evaluate P (s | M), when the path is unknown?***
- ***Can we show the hidden path?***
- ***Can we evaluate the parameters starting from known examples?***

Can we evaluate the parameters starting from known examples?



s :	**A**	**G**	**C**	**G**	**C**	**G**	**T**	**A**	**A**	**T**	**C**	**T**	**G**
π :	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**N**	**N**	**N**	**N**	**N**	**N**

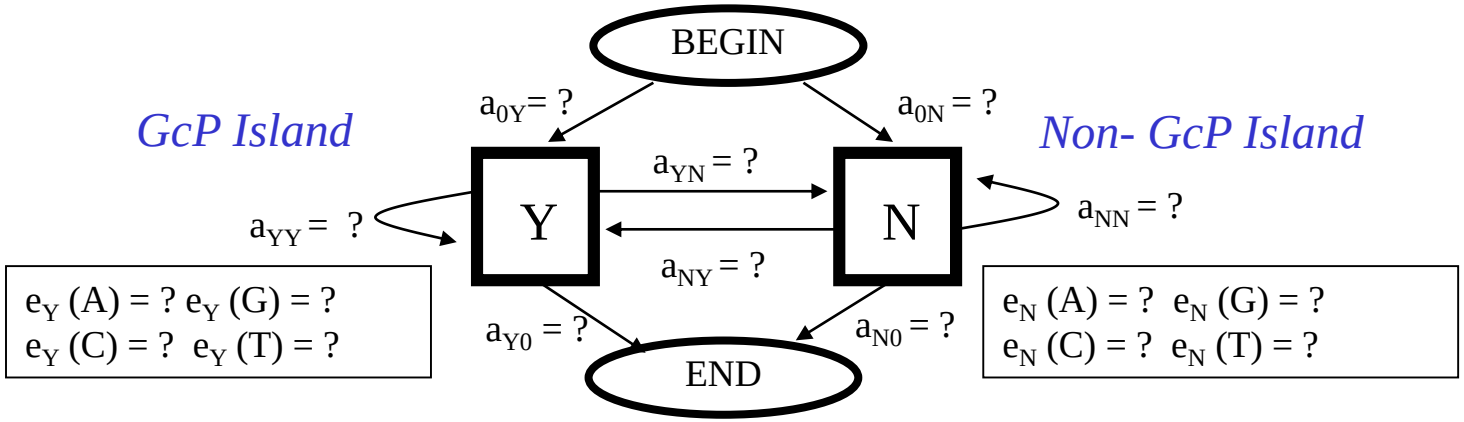
Emission: $e_Y(A) \times e_Y(G) \times e_Y(C) \times e_Y(G) \times e_Y(C) \times e_Y(G) \times e_Y(T) \times e_N(A) \times e_N(A) \times e_N(T) \times e_N(C) \times e_N(T) \times e_N(G)$

Transition: $a_{0Y} \times a_{YY} \times a_{YY} \times a_{YY} \times a_{YY} \times a_{YY} \times a_{YY} \times a_{YN} \times a_{NN} \times a_{NN} \times a_{NN} \times a_{NN} \times a_{NN} \times a_{N0}$

How to find the parameters *e* and *a* that maximises this probability?

How if we don't know the path?

If we know the path generating the training sequence



s :	**A**	**G**	**C**	**G**	**C**	**G**	**T**	**A**	**A**	**T**	**C**	**T**	**G**
π :	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**N**	**N**	**N**	**N**	**N**	**N**

Emission: $e_Y(A) \times e_Y(G) \times e_Y(A) \times e_Y(G) \times e_Y(C) \times e_Y(G) \times e_Y(T) \times e_N(A) \times e_N(A) \times e_N(T) \times e_N(C) \times e_N(T) \times e_N(G)$

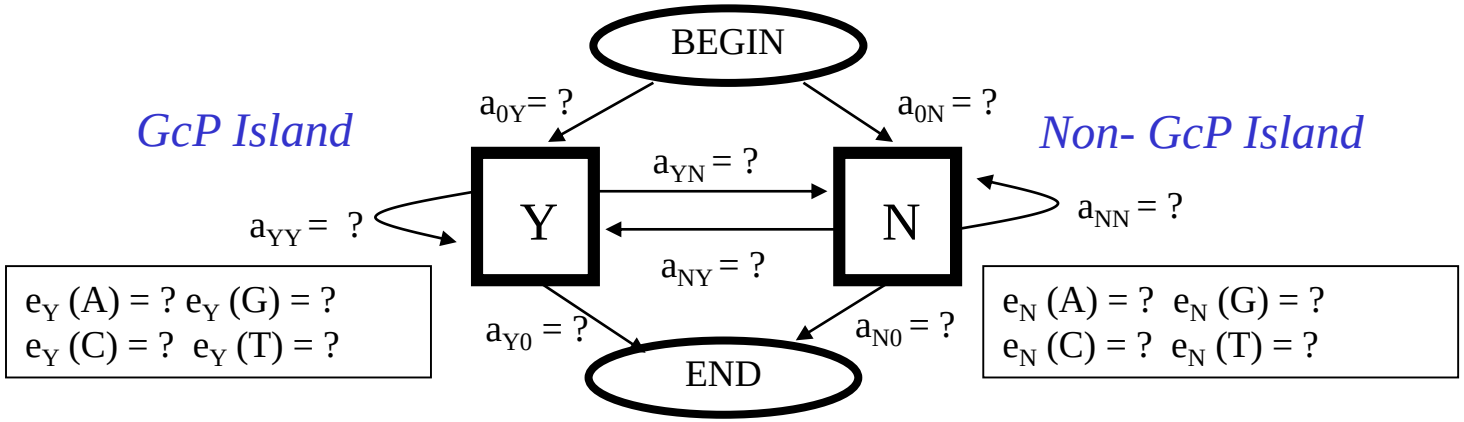
Transition: $a_{0Y} \times a_{YY} \times a_{YY} \times a_{YY} \times a_{YY} \times a_{YY} \times a_{YY} \times a_{YN} \times a_{NN} \times a_{NN} \times a_{NN} \times a_{NN} \times a_{NN} \times a_{N0}$

Just count!

Example: $a_{YY} = n_{YY} / (n_{YY} + n_{YN}) = 6/7$

$e_Y(A) = n_Y(A) / [n_Y(A) + n_Y(C) + n_Y(G) + n_Y(T)] = 1/7$

If we DO NOT know the path generating the training sequence



s :	**A**	**G**	**C**	**G**	**C**	**G**	**T**	**A**	**A**	**T**	**C**	**T**	**G**
π :	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**	**?**

Emission: $e_?(A) \times e_?(G) \times e_?(A) \times e_?(G) \times e_?(C) \times e_?(G) \times e_?(T) \times e_?(A) \times e_?(A) \times e_?(T) \times e_?(C) \times e_?(T) \times e_?(G)$

Transition: $a_{0?} \times a_{??} \times a_{??} \times a_{??} \times a_{??} \times a_{??} \times a_{??} \times a_{??} \times a_{??} \times a_{??} \times a_{??} \times a_{??} \times a_{??} \times a_{?0}$

We need "in some sense" to average over all the possible paths

No exact algorithm is available.

Iterative Baum-Welch algorithm based on the Expectation-Maximisation

Baum-Welch algorithm (simple discussion)

s :	**A**	**G**	**C**	**G**	**C**	**G**	**T**	**A**	**A**	**T**	**C**	**T**	**G**
π_1:	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**
π_2:	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**N**
π_3:	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**N**	**Y**
π_4:	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**N**	**N**

……

Given a path π we can count

the number of transition between states k and l: $A_{k,l}(\pi)$

the number on emissions of character c from state k: $E_k(c,\pi)$

We can compute the expected values over all the paths, given inital parameters θ^0

$$A_{k,l} = \sum_\pi \mathrm{P}(\pi \mid s, \theta^0) \cdot A_{k,l}(\pi) \qquad E_k(c) = \sum_\pi \mathrm{P}(\pi \mid s, \theta^0) \cdot E_k(c,\pi)$$

The updated parameters are:

$$a_{k,l} = \frac{A_{k,l}}{\sum_{m=1}^{N} A_{k,m}} \qquad e_k(c) = \frac{E_k(c)}{\sum_c E_k(c)}$$

Then we can iterate…

Baum-Welch implementation

How to compute the expected number of transitions and emissions over all the paths

$$F_k(i) = P(s^1 s^2 s^3 \ldots\ldots s^i, \pi(i) = k)$$

$$B_k(i) = P(s^{i+1} s^{i+2} s^{i+3} \ldots\ldots s^T \mid \pi(i) = k)$$

$$A_{k,l} = \sum_i P(\pi(i) = k, \pi(i+1) = l \mid s, \theta) = \frac{\sum_i F_k(i) \cdot a_{kl} \cdot e_l(s^{i+1}) \cdot B_l(i+1)}{P(s)}$$

$$E_k(c) = \sum_i P(s^i = c, \pi(i) = k \mid s, \theta) = \frac{\sum_{s^i = c} F_k(i) \cdot B_k(i)}{P(s)}$$

WHY? Expectation-Maximisation algorithm

We need to estimate the Maximum Likelihood parameters when the paths generating the training sequences are unknown

$$\theta^{ML} = \text{argmax}_{\theta} [P (s \mid \theta, M)]$$

Given a model with parameters θ^0 the EM algorithm finds new parameters θ that increase the likelihood of the model:

$$P(s \mid \theta) > P(s \mid \theta^0)$$

WHY? Expectation-Maximisation algorithm

We need to estimate the Maximum Likelihood parameters when the paths generating the training sequences are unknown

$$\theta^{ML} = \text{argmax}_{\theta} [P (s \mid \theta, M)]$$

Given a model with parameters θ^0 the EM algorithm finds new parameters θ that increase the likelihood of the model:

$$P(s \mid \theta) > P(s \mid \theta^0)$$

or equivalentely

$$\ln P(s \mid \theta) > \ln P(s \mid \theta^0)$$

WHY? Expectation-Maximisation algorithm

$$\ln P(s \mid \theta) = \ln P(s,\pi \mid \theta) - \ln P(\pi \mid s,\theta)$$

Multiplying for $P(\pi \mid s,\theta^0)$ and summing over all the possible paths:

$$\ln P(s \mid \theta) =$$

$$= \sum_\pi P(\pi \mid s,\theta^0) \cdot \ln P(s,\pi \mid \theta) - \sum_\pi P(\pi \mid s,\theta^0) \cdot \ln P(\pi \mid s,\theta)$$

$Q(\theta \mid \theta^0)$: Expectation value of $\log P(s,\pi \mid \theta)$ over all the "current" paths

$$\ln P(s \mid \theta) - \ln P(s \mid \theta^0) =$$

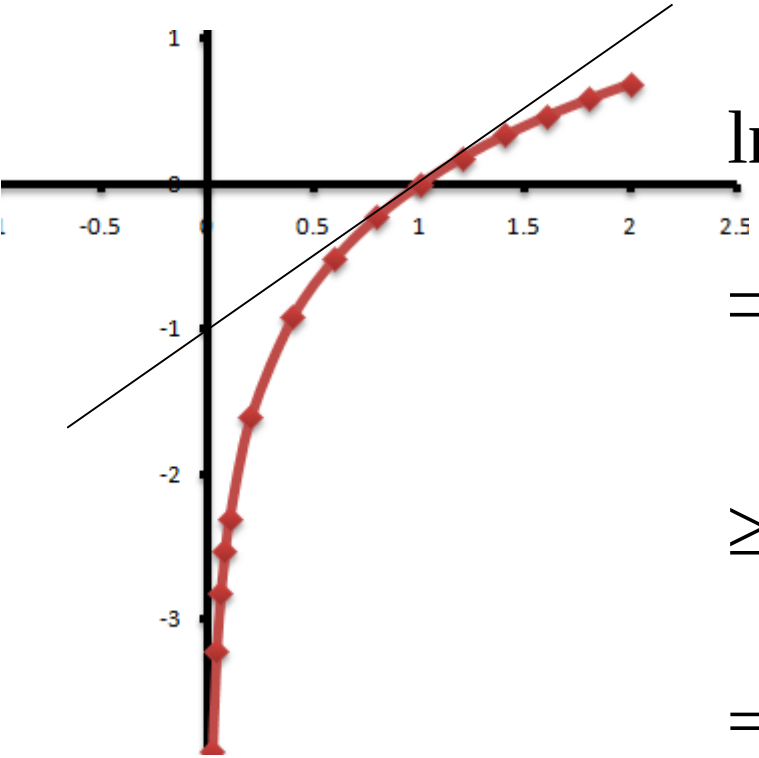
$$= Q(\theta \mid \theta^0) - Q(\theta^0 \mid \theta^0) + \sum_\pi - P(\pi \mid s,\theta^0) \cdot \ln \frac{P(\pi \mid s,\theta)}{P(\pi \mid s,\theta^0)}$$

WHY? Expectation-Maximisation algorithm

$$\ln P(s \mid \theta) - \ln P(s \mid \theta^0) =$$

$$= Q(\theta \mid \theta^0) - Q(\theta^0 \mid \theta^0) + \underbrace{\sum_\pi - P(\pi \mid s, \theta^0) \cdot \ln \frac{P(\pi \mid s, \theta)}{P(\pi \mid s, \theta^0)}}_{\geq 0}$$

$$\geq Q(\theta \mid \theta^0) - Q(\theta^0 \mid \theta^0)$$



$$\ln x \leq x - 1 \Rightarrow$$

$$\Rightarrow \sum_\pi - P(\pi \mid s, \theta^0) \cdot \ln \frac{P(\pi \mid s, \theta)}{P(\pi \mid s, \theta^0)} \geq$$

$$\geq \sum_\pi - P(\pi \mid s, \theta^0) \cdot \left(\frac{P(\pi \mid s, \theta)}{P(\pi \mid s, \theta^0)} - 1 \right) =$$

$$= \sum_\pi \left[- P(\pi \mid s, \theta) + P(\pi \mid s, \theta^0) \right] =$$

$$= -\sum_\pi P(\pi \mid s, \theta) + \sum_\pi P(\pi \mid s, \theta^0) = 0$$

WHY? Expectation-Maximisation algorithm

The EM algorithm is an iterative process

Each iteration performs two steps:

E-step: evaluation of $Q(\theta|\theta^0) = \sum_\pi P(\pi|s,\theta^0) \cdot \log P(s,\pi|\theta)$

M-step: Maximisation of $Q(\theta|\theta^0)$ over all θ

It does NOT assure to converge to the GLOBAL Maximum Likelihood

Baum-Welch implementation of the EM algorithm

E-step:

$$Q(\theta|\theta^0) = \sum_\pi P(\pi|s,\theta^0) \cdot \log P(s,\pi|\theta)$$

$$P(s,\pi|\theta) = a_{0,\pi(1)} \cdot \prod_{i=1}^{T} a_{\pi(i),\pi(i+1)} \cdot e_{\pi(i)}(s^i) =$$

$$= \prod_{k=0}^{N} \prod_{l=1}^{N} a_{k,l}^{A_{k.l}(\pi)} \cdot \prod_{k=1}^{N} \prod_{c \in C} e_k(c)^{E_k(c,\pi)}$$

$A_{k,l}(\pi)$: number of transitions between the states k and l in path π

$E_k(c,\pi)$: number of emissions of character c in path π

$$A_{k,l} = \sum_\pi P(\pi|s,\theta^0) \cdot A_{k,l}(\pi)$$

$$E_k(c) = \sum_\pi P(\pi|s,\theta^0) \cdot E_k(c,\pi)$$

Expected values over all the "actual" paths

So:

$$Q(\theta|\theta^0) = \sum_{k=0}^{N} \sum_{l=1}^{N} A_{k,l} \cdot \ln a_{k,l} + \sum_{k=1}^{N} \sum_{c \in C} E_k(c) \cdot \ln e_k(c)$$

Baum-Welch implementation of the EM algorithm

M-step:

$$\frac{\partial Q}{\partial a_{k,l}} = 0$$ For any state k and l, with $\sum_l a_{k,l} = 1$

$$\frac{\partial Q}{\partial e_k(c)} = 0$$ For any state k and character c, with $\sum_c e_k(c) = 1$

By means of Lagrange's multipliers techniques, we can solve the system

$$L(a,e,\lambda,\mu) = \sum_{k=0}^{N}\sum_{l=1}^{N} A_{kl}\ln a_{kl} + \sum_{k=0}^{N}\sum_{c\in C} E_k(c)\ln e_k(c) - \sum_{i=1}^{N}\lambda_i\left(\sum_{j=1}^{N} a_{ij} - 1\right) - \sum_{i=1}\mu_i\left(\sum_{c\in C} e_i(c) - 1\right)$$

2 Lagrange's multipliers for each state are introduced

Baum-Welch implementation of the EM algorithm

M-step:

$$L(a,e,\lambda,\mu)=\sum_{k=0}^{N}\sum_{l=1}^{N}A_{kl}\ln a_{kl}+\sum_{k=0}^{N}\sum_{c\in C}E_k(c)\ln e_k(c)-\sum_{i=1}^{N}\lambda_i\left(\sum_{j=1}^{N}a_{ij}-1\right)-\sum_{i=1}\mu_i\left(\sum_{c\in C}e_i(c)-1\right)$$

$$\begin{cases}\dfrac{\partial L}{\partial a_{kl}}=0,\forall k,l\\[2ex] \dfrac{\partial L}{\partial e_k(c)}=0,\forall k,c\\[2ex] \dfrac{\partial L}{\partial \lambda_k}=0,\forall k\\[2ex] \dfrac{\partial L}{\partial \mu_k}=0,\forall k\end{cases}\Rightarrow\begin{cases}\dfrac{A_{kl}}{a_{kl}}-\lambda_k=0,\forall k,l\\[2ex] \dfrac{E_k(c)}{e_k(c)}-\mu_k=0,\forall k,c\\[2ex] \displaystyle\sum_{j=1}^{N}a_{kj}=1,\forall k\\[2ex] \displaystyle\sum_{c\in C}e_k(c)=1,\forall k\end{cases}$$

Baum-Welch implementation of the EM algorithm

M-step:

$$L(a,e,\lambda,\mu)=\sum_{k=0}^{N}\sum_{l=1}^{N}A_{kl}\ln a_{kl}+\sum_{k=0}^{N}\sum_{c\in C}E_k(c)\ln e_k(c)-\sum_{i=1}^{N}\lambda_i\left(\sum_{j=1}^{N}a_{ij}-1\right)-\sum_{i=1}^{N}\mu_i\left(\sum_{c\in C}e_i(c)-1\right)$$

$$\begin{cases}\dfrac{A_{kl}}{a_{kl}}-\lambda_k=0,\forall k,l\\[2ex] \dfrac{E_k(c)}{e_k(c)}-\mu_k=0,\forall k,c\\[2ex] \sum_{j=1}^{N}a_{kj}=1,\forall k\\[2ex] \sum_{c\in C}e_k(c)=1,\forall k\end{cases}\Rightarrow\begin{cases}a_{kl}=\dfrac{A_{kl}}{\lambda_k}\forall k,l\\[2ex] e_k(c)=\dfrac{E_k(c)}{\mu_k},\forall k,c\\[2ex] \sum_{j=1}^{N}a_{kj}=1,\forall k\\[2ex] \sum_{c\in C}e_k(c)=1,\forall k\end{cases}$$

Baum-Welch implementation of the EM algorithm

M-step:

$$a_{kl} = \frac{A_{kl}}{\sum_{i=1}^{N} A_{ki}}$$

$$e_k(c) = \frac{E_k(c)}{\sum_{d \in C} E_k(d)}$$

Baum-Welch implementation of the EM algorithm

How to compute the expected number of transitions and emissions over all the paths

$F_k(i) = P(\, s^1 s^2 s^3 \ldots\ldots s^{\,i},\ \pi(i) = k\,)$

$B_k(i) = P(\, s^{\,i+1} s^{\,i+2} s^{\,i+3} \ldots\ldots s^{\,T} \mid \pi(i) = k\,)$

$$A_{kl} = \sum_i P\left(\pi_i = k, \pi_{i+1} = l \mid s\right) = \frac{\sum_i F_k(i) \cdot a_{kl} \cdot e_l(s^{i+1}) \cdot B_l(i+1)}{P(s)}$$

$$E_k(c) = \sum_{s^i = c} P\left(s^i = c, \pi_i = k \mid s\right) = \frac{\sum_{s^i = c} F_k(i) \cdot B_k(i)}{P(s)}$$

This is for 1 training sequence: for general training with many sequences, just sum over them

Baum-Welch implementation of the EM algorithm:proof

$$A_{kl} = \sum_i P(\pi_i = k, \pi_{i+1} = l \mid s) = \frac{P(s, \pi_i = k, \pi_{i+1} = l)}{P(s)} =$$

$$= \frac{\sum_i P(s^1, s^2..s^i, \pi_i = k) \cdot P(\pi_{i+1} = l \mid \pi_i = k) \cdot P(s^{i+1} \mid \pi_{i+1} = l) \cdot P(s^{i+2},..s^T \mid \pi_{i+1} = l)}{P(s)} =$$

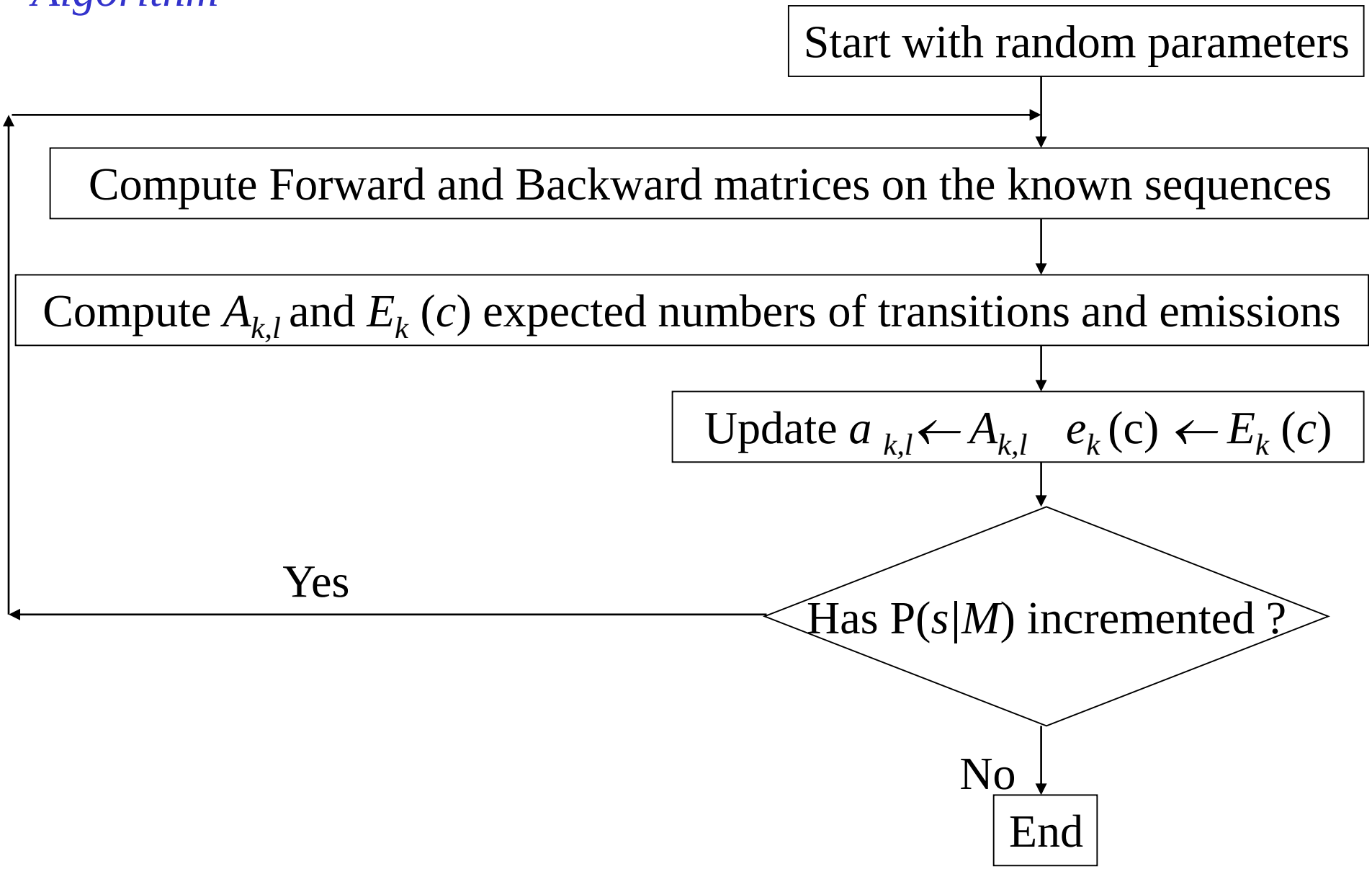
$$= \frac{\sum_i F_k(i) \cdot a_{kl} \cdot e_l(s^{i+1}) \cdot B_l(i+1)}{P(s)}$$

$$E_k(c) = \sum_{s^i = c} P(s^i = c, \pi_i = k \mid s) = \frac{\sum_{s^i = c} P(s^1, s^2.., s^i = c, s^{i+1},.., s^T, \pi_i = k)}{P(s)} =$$

$$= \frac{\sum_{s^i = c} P(s^1, s^2.., s^i = c, \pi_i = k) \cdot P(s^{i+1},.., s^T \mid \pi_i = k)}{P(s)} = \frac{\sum_{s^i = c} F_k(i) \cdot B_k(i)}{P(s)}$$

Baum-Welch implementation of the EM algorithm

Algorithm



Profile HMMs

- HMMs for alignments

PROLOGUE:

Pitfalls of standard alignments

Scoring a pairwise alignment

A: ALA**E**VLIRLIT**K**LYP

B: ASA**K**HLNRLIT**E**LYP

$$Score(A,B) = \sum s(A^i, B^i)$$

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

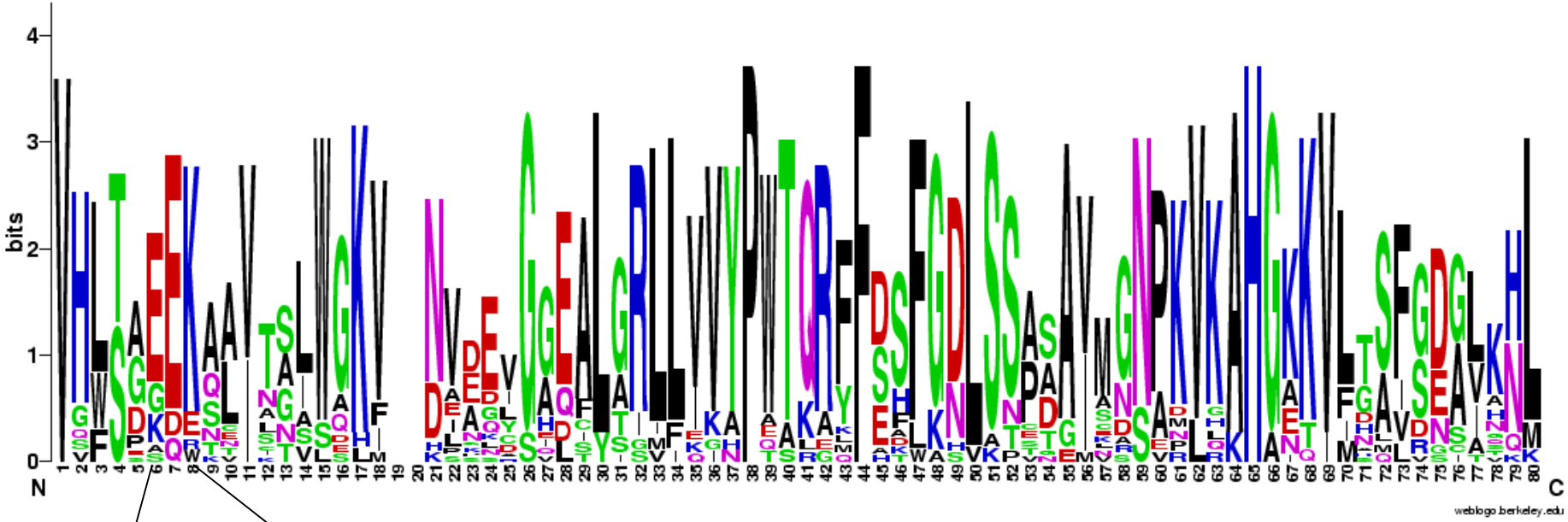
Blosum62

Alignment of a family (globins)

```
                                 10        20        30        40        50        60        70        80
lqb1 pea/1-471   -GFTDKQEALVNSSSE-FKQNLPGYSILFYTIVLEKAPAAKGLFSFLKD---TAGVEDSPKLQAHAEQVFGLVRDSAAQL
lqb1 vicfa/1-471-GFTEKQEALVNSSSQLFKQNPSNYSVLFYTIILQKAPTAKAMFSFLKD---SAGVVDSPKLGAHAEKVFGMVRDSAVQL
hbb speci/1-471 VHLSDGEKNAISTAWGKV--HAAEVGAEALGRLLVVYPWTQRFFDSFGDLSSASAVMGNAKVKAHGKKVIDSFSNGLKHL
hbb speto/1-471 VHLTDGEKNAISTAWGKV--NAAEIGAEALGRLLVVYPWTQRFFDSFGDLSSASAVMGNAKVKAHGKKVIDSFSNGLKHL
hbb equhe/1-471 VQLSGEEKAAVLALWDKV--NEEEVGGEALGRLLVVYPWTQRFFDSFGDLSNPAAVMGNPKVKAHGKKVLHSFGEGVHHL
hbb sunmu/1-471 VHLSGEEKACVTGLWGKV--NEDEVGAEALGRLLVVYPWTQRFFDSFGDLSSASAVMGNPKVKAHGKKVLHSLGEGVANL
hbb tupql/1-471 VHLSGEEKAAVTGLWGKV--DLEKVGGQSLGSLLIVYPWTQRFFDSFGDLSSPSAVMSNPKVKAHGKKVLTSFSDGLNHL
hbb calar/1-471 VHLTGEEKSAVTALWGKV--NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMNNPKVKAHGKKVLGAFSDGLTHL
hbb mansp/1-471 VHLTPEEKTAVTTLWGKV--NVDEVGGEALGRLLVVYPWTQRFFDSFGDLSSPDAVMGNPKVKAHGKKVLGAFSDGLNHL
hbb rabit/1-471 VHLSSEEKSAVTALWGKV--NVEEVGGEALGRLLVVYPWTQRFFESFGDLSSANAVMNNPKVKAHGKKVLAAFSEGLSHL
hbb ursma/1-471 VHLTGEEKSLVTGLWGKV--NVDEVGGEALGRLLVVYPWTQRFFDSFGDLSSADAIMNNPKVKAHGKKVLNSFSDGLKNL
hbb triin/1-471 VHLTPEEKALVIGLWAKV--NVKEYGGEALGRLLVVYPWTQRFFEHFGDLSSASAIMNNPKVKAHGEKVFTSFGDGLKHL
hbb ornan/1-471 VHLSGGEKSAVTNLWGKV--NINELGGEALGRLLVVYPWTQRFFEAFGDLSSAGAVMGNPKVKAHGAKVLTSFGDALKNL
hbb tacac/1-471 VHLSGSEKTAVTNLWGHV--NVNELGGEALGRLLVVYPWTQRFFESFGDLSSADAVMGNAKVKAHGAKVLTSFGDALKNL
hbe ponpy/1-471 VHFTAEEKAAVTSLWSKM--NVEEAGGEALGRLLVVYPWTQRFFDSFGNLSSPSAILGNPKVKAHGKKVLTSFGDAIKNM
hbb colli/1-471 VHWSAEEKQLITSIWGKV--NVADCGAEALARLLIVYPWTQRFFSSFGNLSSATAISGNPNVKAHGKKVLTSFGDAVKNL
hbb larri/1-471 VHWSAEEKQLITGLWGKV--NVADCGAEALARLLIVYPWTQRFFASFGNLSSPTAINGNPMVRAHGKKVLTSFGEAVKNL
hbb1 varex/1-471VHWTAEEKQLICSLWGKI--DVGLIGGETLAGLLVIYPWTQRQFSHFGNLSSPTAIAGNPRVKAHGKKVLTSFGDAIKNL
hbb2 xentr/1-471VHWTAEEKATIASVWGKV--DIEQDGHDALSRLLVVYPWTQRYFSSFGNLSNVSAVSGNVKVKAHGNKVLSAVGSAIQHL
hbb1 ranca/1-471VHWTAEEKAVINSVWQKV--DVEQDGHEALTRLFIVYPWTQRYFSTFGDLSSPAAIAGNPKVHAHGKKILGAIDNAIHNL
hbb2 tricr/1-471VHLTAEDRKEIAAILGKV--NVDSLGGQCLARLIVVNPWSRRYFHDFGDLSSCDAICRNPKVLAHGAKVMRSIVEATKHL
hba4 salir/1-471-SLSAKDKANVKAIWGKILPKSDEIGEQALSRMLVVYPQTKAYFSHWASVAP-----GSAPVKKHGITIMNQIDDCVGHM
myg_escgi/1-471 -VLSDAEWQLVLNIWAKVEADVAGHGQDILIRLFKGHPETLEKFDKFKHLKTEAEMKASEDLKKHGNTVLTALGGILKKK
```

Different positions are not equivalent

Sequence logos

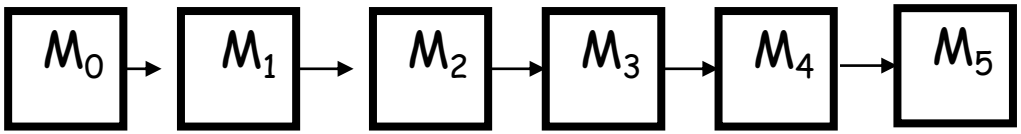


http://weblogo.berkeley.edu/cache/file5h2DWc.png

The substitution score IN A FAMILY should depend on the position (the same for gaps)

For modelling families we need more flexible tools

How to align?



Each state represent a position in the alignment.

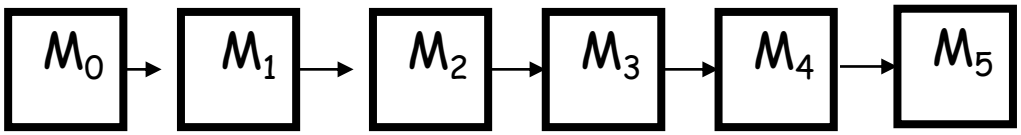
A	C	G	G	T	A
M_0	M_1	M_2	M_3	M_4	M_5
A	C	G	A	T	C
M_0	M_1	M_2	M_3	M_4	M_5
A	T	G	T	T	C
M_0	M_1	M_2	M_3	M_4	M_5

Each position has a peculiar composition

Given a set of sequences..

A	C	G	G	T	A
A	C	G	A	T	C
A	T	G	T	T	C

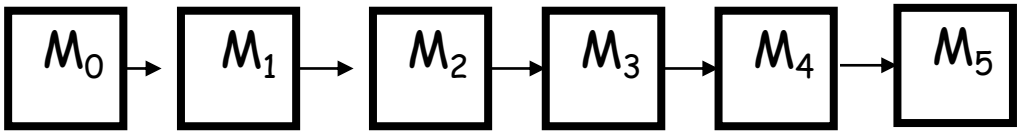
..we can train a model..



..estimating the emission probabilities.

A	1	0	0	0.33	0	0.33
C	0	0.66	0	0	0	0.66
G	0	0	1	0.33	0	0
T	0	0.33	0	0.33	1	0

Given a trained model..



A	1	0	0	0.33	0	0.33
C	0	0.66	0	0	0	0.66
G	0	0	1	0.33	0	0
T	0	0.33	0	0.33	1	0

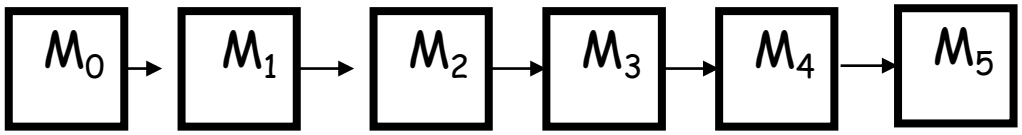
..we can align a new sequence..

A C G A T C

..computing the probability of generating it

$$P(s|M) = 1 \times 0.66 \times 1 \times 0.33 \times 1 \times 0.66$$

And for the sequence AGATC ?

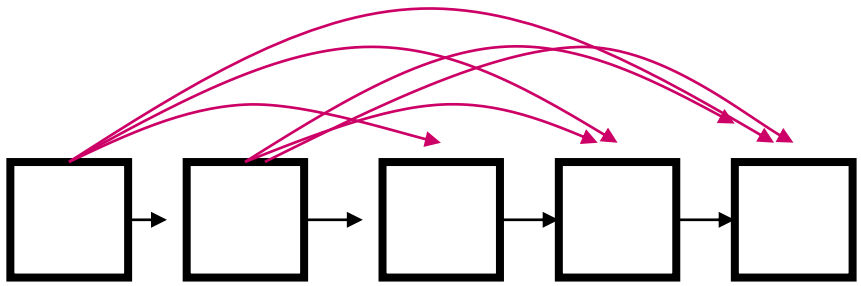


A	1	0	0	0.33	0	0.33
C	0	0.66	0	0	0	0.66
G	0	0	1	0.33	0	0
T	0	0.33	0	0.33	1	0

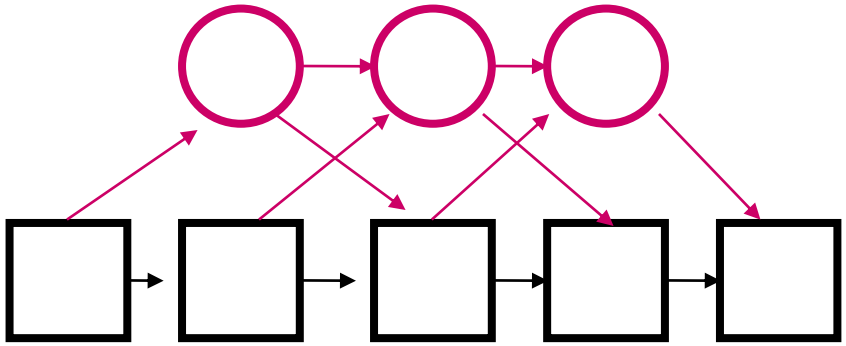
A		G	A	T	C
M_0	M_2	M_3	M_4	M_5	M_5

We need a way to introduce gaps

Silent states

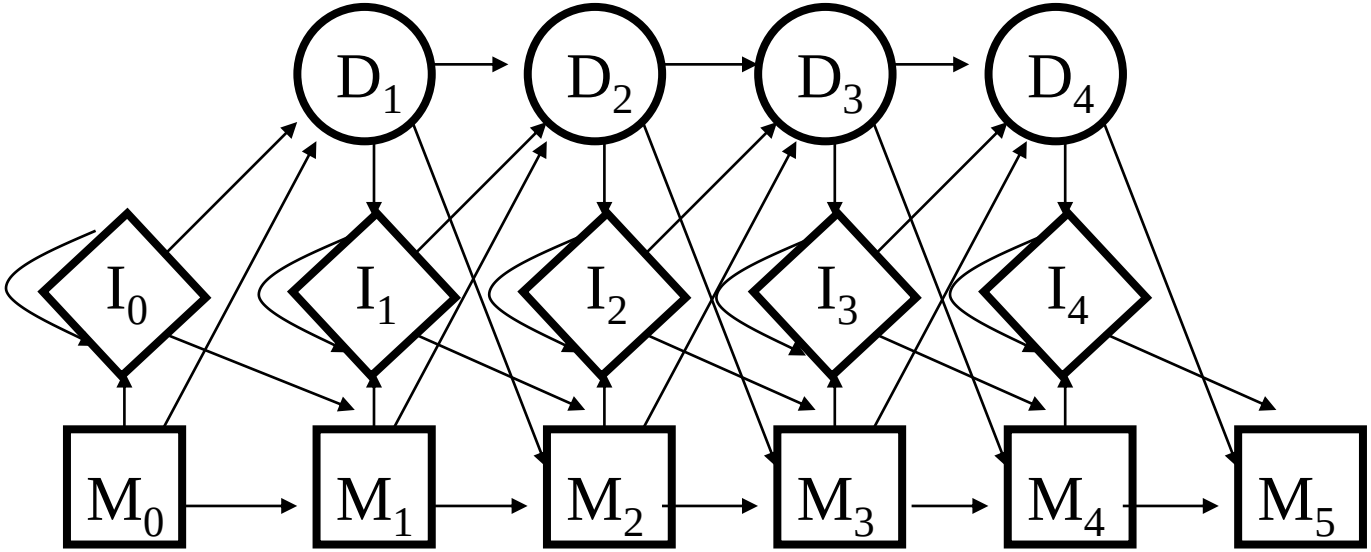


Red transitions allow gaps
N(N-1)/2 transitions



To reduce the number of parameters we can use states that **doesn't emit any character**
4N-8 transitions

Profile HMMs



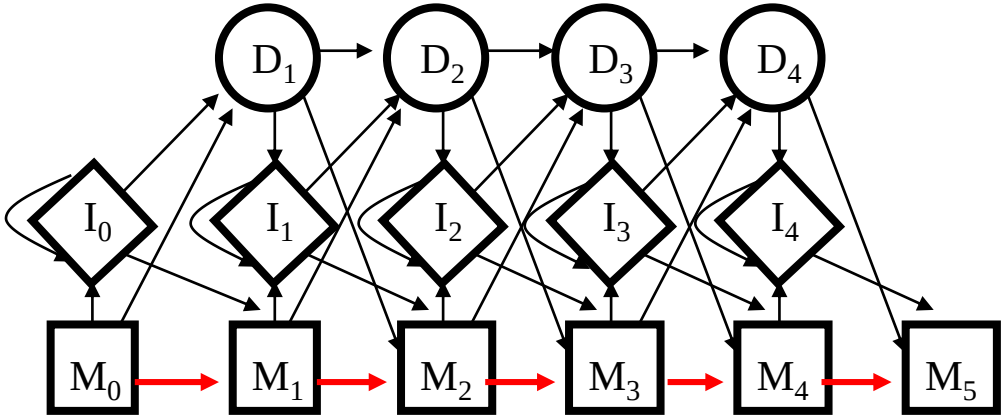
Delete states

Insert states

Match states

A	C	G	G	T	A
M_0	M_1	M_2	M_3	M_4	M_5
A C G	C	A	G	T	C
M_0 I_0 I_0	M_1	M_2	M_3	M_4	M_5
A		G	A	T	C
M_0	D_1	M_2	M_3	M_4	M_5

Example of alignment



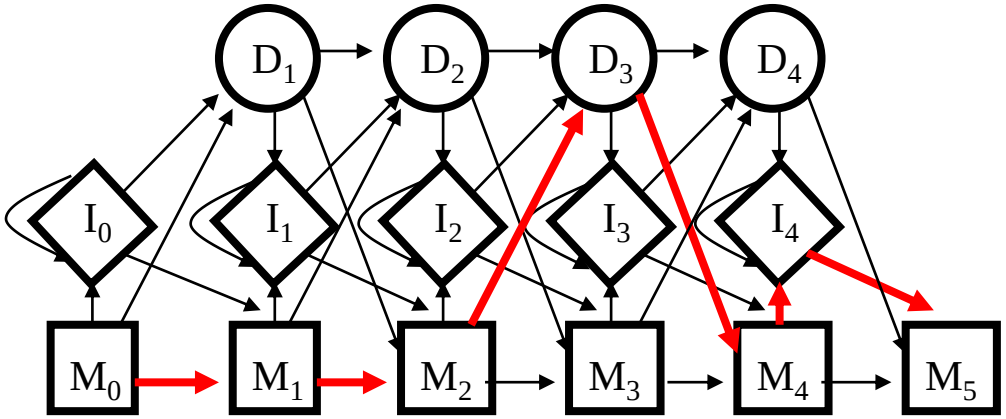
Sequence 1

A S T R A L

Viterbi path

M_0 M_1 M_2 M_3 M_4 M_5

A S T R A L



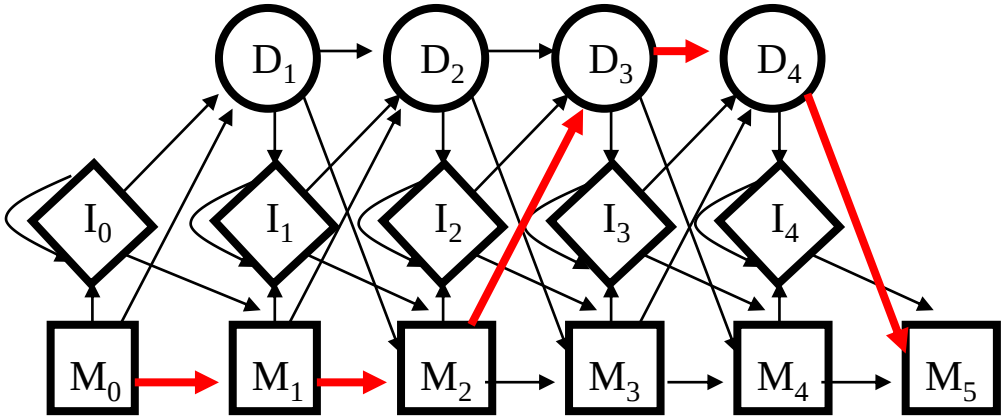
Sequence 2

A S T A I L

Viterbi path

M_0 M_1 M_2 D_3 M_4 I_4 M_5

A S T A I L



Sequence 3

A R T I

Viterbi path

M_0 M_1 M_2 D_3 D_4 M_5

A R T I

Example of alignment

M_0	M_1	M_2	M_3	M_4		M_5
A	S	T	R	A		L
M_0	M_1	M_2	D_3	M_4	I_4	M_5
A	S	T		A	I	L
M_0	M_1	M_2	D_3	D_4		M_5
A	R	T				I

Sequence 1

Sequence 2

Sequence 3

Grouping by vertical layers

	0	1	2	3	4	5
s_1	A	S	T	R	A	L
s_2	A	S	T		AI	L
s_3	A	R	T			I

Alignment

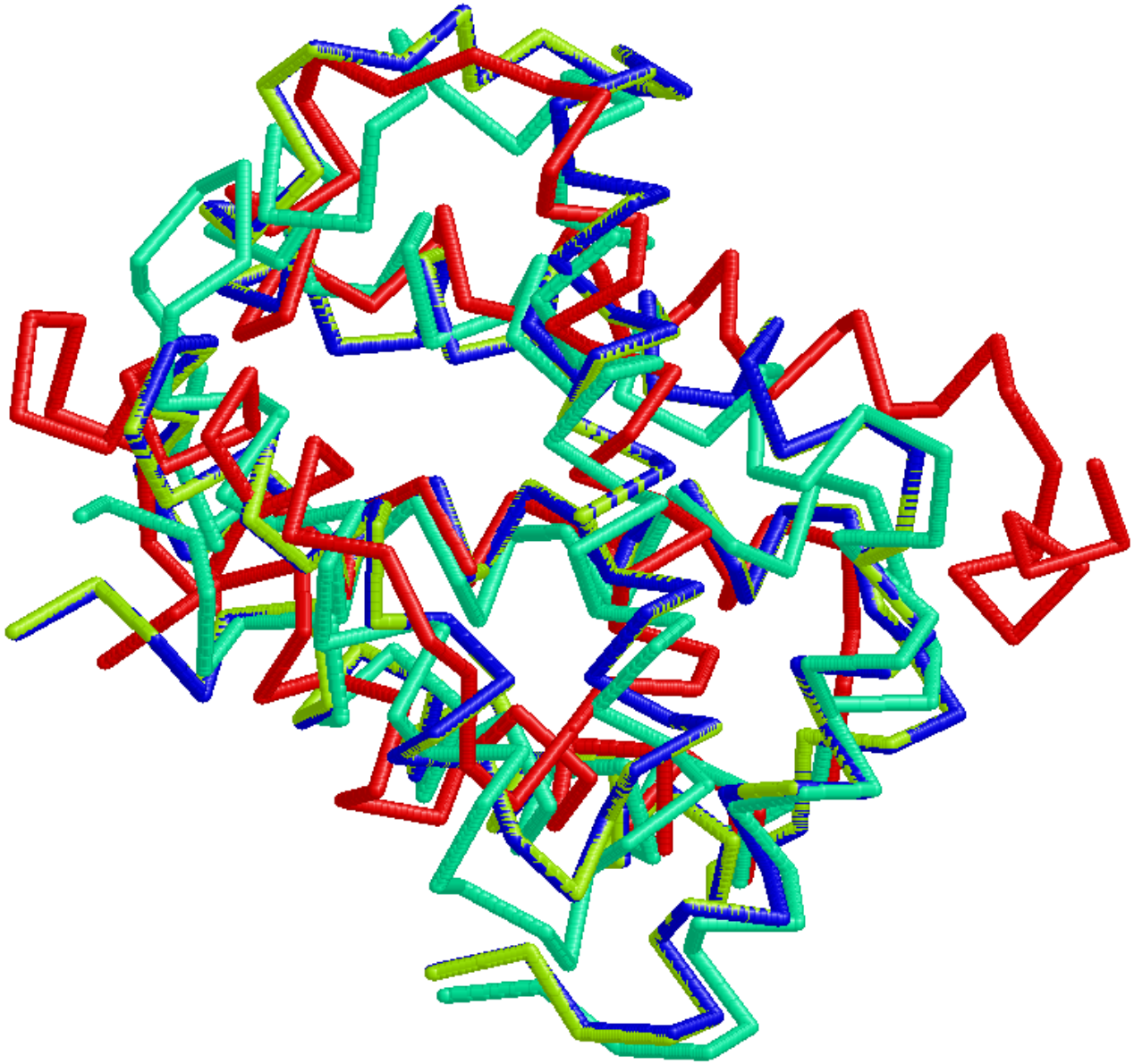
```
ASTRA-L
AST-AIL
ART---I
```

$-\text{Log } P(s \mid M)$ Is an alignment score

Profile HMMs

- HMMs for alignments
- Example on globins

Structural alignment of globins



Structural alignment of globins

```
            AAAAAAAAAAAAAAAA   BBBBBBBBBBBBBBBBCCCCCCCCCCC
                                                   DDDD
---------VLSPADKTNVKAAWGKVGA--HAGEYGAEALERMFLSFPTTKTYFPHF-DL
--------VHLTPEEKSAVTALWGKV----NVDEVGGEALGRLLVVYPWTQRFFESFGDL
---------VLSEGEWQLVLHVWAKVEA--DIAGHGQDILIRLFKHHPETLEKFDRFKHL
----------LSADQISTVQASFDKVKG------DPVGILYAVFKADPSIMAKFTQFAG-
PIVDTGSVAPLSAAEKTKIRSAWAPVYS--TYETSGVDILVKFFTSTPAAQEFFPKFKGL
--------GALTESQAALVKSSWEEFNA--NIPKHTHRFFILVLEIAPAAKDLFS-FLK-
---------GLSAAQRQVIAATWKDIAGADNGAGVGKDCLIKFLSAHPQMAAVFG-FSG-


 DDDDDDDEEEEEEEEEEEEEEEEEEEE            FFFFFFFFFFFF  FFGGG
                            F                     GG  GG
S-----HGSAQVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKL--RVDPV
STPDAVMGNPKVKAHGKKVLGAFSDGLAHL---D--NLKGTFATLSELHCDKL--HVDPE
KSEAEMKASEDLKKHGVTVLTALGAILKK----K-GHHEAELKPLAQSHATKH--KIPIK
KDLESIKGTAPFETHANRIVGFFSKIIGEL--P---NIEADVNTFVASHKPRG---VTHD
TTADQLKKSADVRWHAERIINAVNDAVASM--DDTEKMSMKLRDLSGKHAKSF--QVDPQ
GTSEVPQNNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG---VADA
---AS---DPGVAALGAKVLAQIGVAVSHL--GDEGKMVAQMKAVGVRHKGYGNKHIKAQ


GGGGGGGGGGGGGGG     HHHHHHHHHHHHHHHHHHHHHHHHHH

NFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKYR------
NFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKYH------
YLEFISEAIIHVLHSRHPADFGADAQGAMSKALELFRKDIAAKYKELGYQG
QLNNFRAGFVSYMKAHT--DFA-GAEAAWGATLDTFFGMIFSKM-------
YFKVLAAVIADTVAAG---------DAGFEKLMSMICILLRSAY-------
HFPVVKEAILKTIKEVVGAKWSEELNSAWTIAYDELAIVIKKEMNDAA---
YFEPLGASLLSAMEHRIGGKMNAAAKDAWAAAYADISGALISGLQS-----
```

Bashdorf D, Chothia C & Lesk AM, (1987) *Determinants of a protein fold: unique features of the globin amino sequence*. **J.Mol.Biol. 196**, 199-216

Alignment of globins reconstructed with profile HMMs

The parameters of an HMM are estimated from a training set of 400 unaligned sequences. After the HMM is built, it is used to obtain a multiple alignment of all the training sequences. This is the alignment f the 7 globins as aligned with the trained model

```
             AAAAAAAAAAAAAAAA   BBBBBBBBBBBBBBBBCCCCCCCCCCC
                                                 DDDD
             ***************    ************************
V.........LSPADKTNVKAAWGKVGA..HAGEYGAEALERMFLSFPTTKTYFPHFD-L
Vh........LTPEEKSAVTALWGKV--..NVDEVGGEALGRLLVVYPWTQRFFESFGDL
V.........LSEGEWQLVLHVWAKVEA..DVAGHGQDILIRLFKSHPETLEKFDRFKHL
-.........LSADQISTVQASFDKV--..KGDPVGI--LYAVFKADPSIMAKFTQFAGK
PivdtgsvapLSAAEKTKIRSAWAPVYS..TYETSGVDILVKFFTSTPAAQEFFPKFKGL
Ga........LTESQAALVKSSWEEFNA..NIPKHTHRFFILVLEIAPAAKDLFSFLK-G
G.........LSAAQRQVIAATWKDIAGadNGAGVGKDCLIKFLSAHPQMA---AVFG-F


 DDDDDDDEE EEEEEEEEEEEEEEEEE            FFFFFFFFFF   FFFFG
                            F                        GGGG
                    ***********            *********       **
SHGSAQVKGH-GKK.----VADALTNAVAHVDD.....MPNALSALSDLHA...HKLRVD
STPDAVMGNPKVKA.HGKKVLGAFSDGLAHLDN.....LKGTFATLSELHC...DKLHVD
KTEA-EMKASEDLKkHGVTVLTALGAILKKKGH.....HEAELKPLAQSHA...TKHKIP
DLES-IKGTAPFET.HANRIVGFFSKIIGELPN.....IEADVNTFVASHK...PR-GVT
TTADQLKKSADVRW.HAERIINAVNDAVASMDDtek..MSMKLRDLSGKHA...KSFQVD
TSEVPQ-NNPELQA.HAGKVFKLVYEAAIQLQVtgvvvTDATLKNLGSVHV...SK-GVA
SGAS----DPGVAA.LGAKVLAQIGVAVSHLGDegk..MVAQMKAVGVRHKgygNK-HIK


GGGGGGGGGGGGGGGGGG     HHHHHHHHHHHHHHHHHHHHHHHHHH
******************    ************************
PVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVLTSKY......R
PENFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANALAHKY......H
IKYLEFISEAIIHVLHSRHPGDFGADAQGAMNKALELFRKDIAAKYkelgyqG
HDQLNNFRAGFVSYMKAH--TDF-AGAEAAWGATLDTFFGMIFSKM......-
PQYFKVLAAVIADTVAA---GD------AGFEKLMSMICILLRSAY......-
DAHFPVVKEAILKTIKEVVGAKWSEELNSAWTIAYDELAIVIKKEMnda...A
AQYFEPLGASLLSAMEHRIGGKMNAAAKDAWAAAYADISGALISGLq.....S
```

Krogh A, Brown M, Mian IS, Sjolander K & Haussler D (1994) *Hidden Markov Models in computational biology: applications to protein modelling*. **J.Mol.Biol. 235**, 1501-1531

Discrimination power of profile HMMs

NLL-score= -log P(s|M)

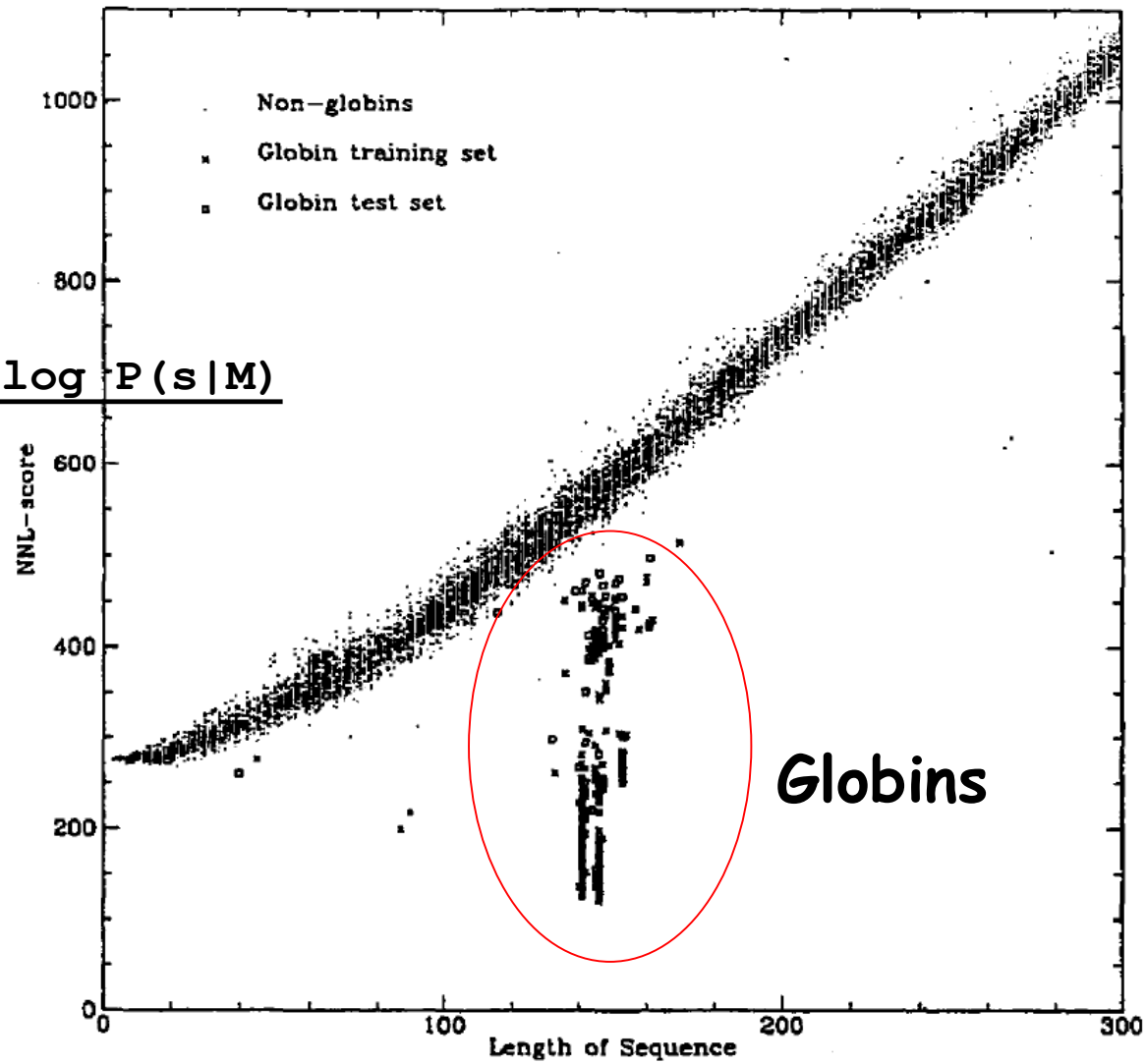


Figure 6. Plot of NLL-score *versus* sequence length for globins and non-globins. All sequences of length less than 300 from the SWISS-PROT 22 database are shown, including partial sequences and 3 false globins from the globin file, and sequences from the database containing many Xs.

Krogh A, Brown M, Mian IS, Sjolander K & Haussler D (1994) *Hidden Markov Models in computational biology: applications to protein modelling*. **J.Mol.Biol. 235**, 1501-1531

HMMs for Mapping problems

- Mapping problems in protein prediction

Secondary structure

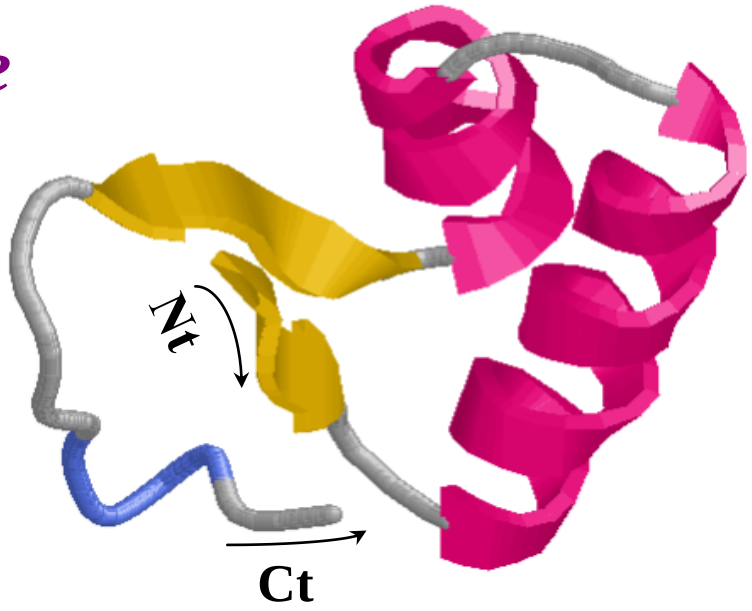
Covalent structure

TTCCPSIVARSNFNVCRLPGTPEAICATYTGCIIIPGATCPGDYAN

Secondary structure

EEEE..HHHHHHHHHHHH....HHHHHHHH.EEEE.....

3D structure

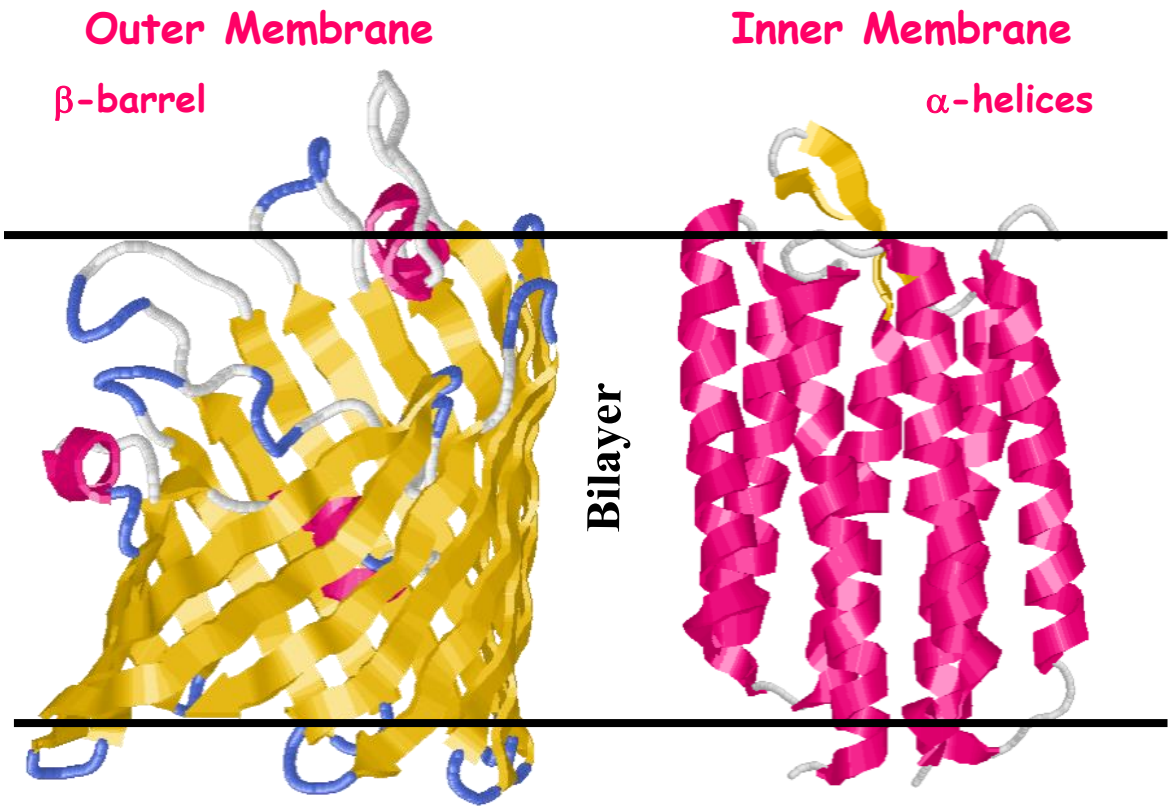


Topology of membrane proteins

Topography

position of Trans Membrane Segments along the sequence

ALALMLCMLTYRHKELKLKLKK ALALMLCMLTYRHKELKLKLKK ALALMLCMLTYRHKELKLKLKK



Porin
(*Rhodobacter capsulatus*)

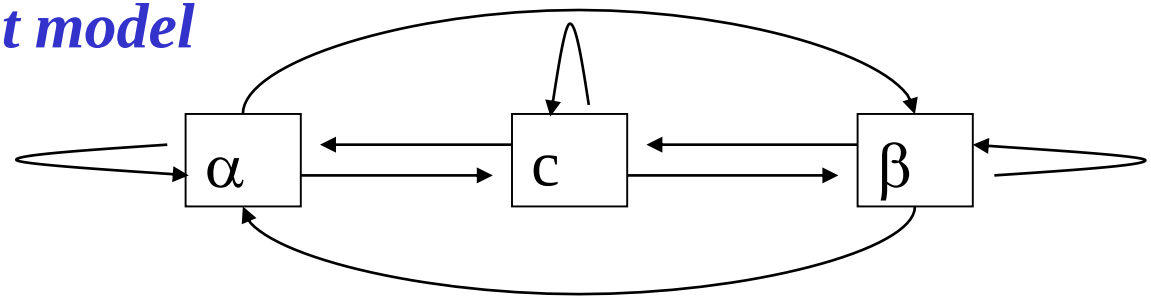
Bacteriorhodopsin
(*Halobacterium salinarum*)

HMMs for Mapping problems

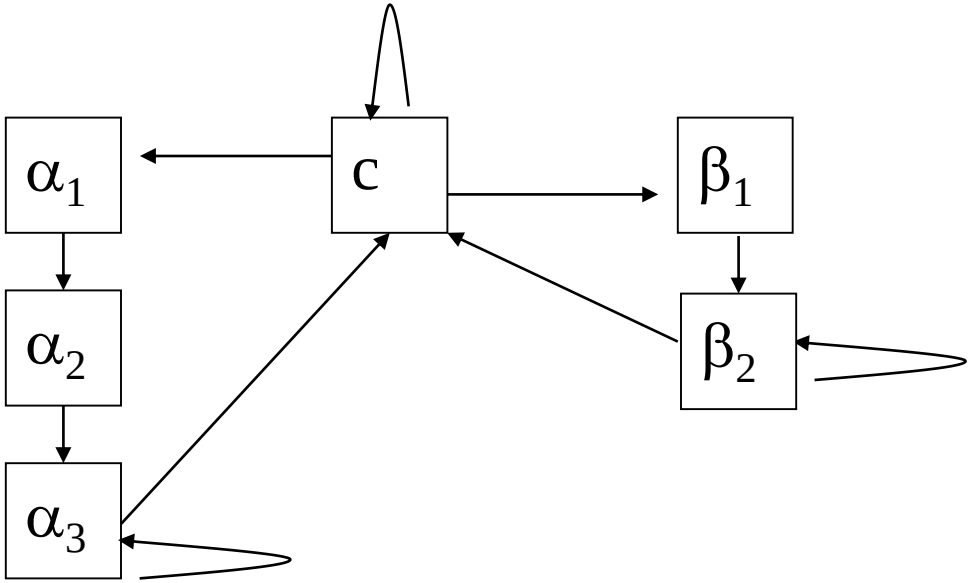
- Mapping problems in protein prediction
- Labelled HMMs

HMM for secondary structure prediction

Simplest model



Introducing a grammar

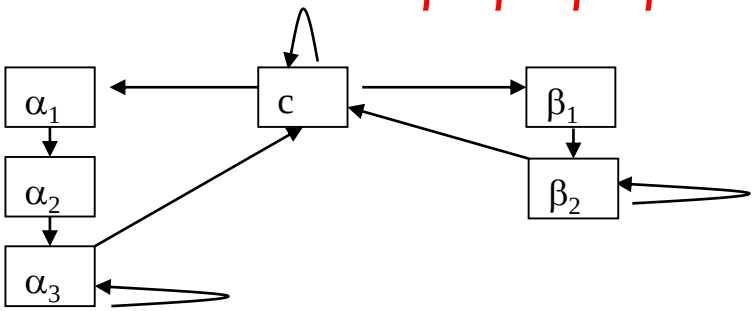


HMM for secondary structure prediction

Labels

The states α_1, α_2 and α_3 share the same *label*, so states β_1 and β_2 do.
Decoding the
Viterbi path for emitting a sequence *s*, makes a mapping between the sequence *s* and a sequence of labels *y*

S	**A**	**L**	**K**	**M**	**N**	**Y**	**T**	**R**	**E**	**I**	**M**	**V**	**A**	**S**	**N**	**Q**	*s*: *Sequence*
c	α_1	α_2	α_3	α_4	α_4	α_4	c	c	c	c	β_1	β_2	β_2	β_2	c	c	π: *Path*
c	α	α	α	α	α	α	c	c	c	c	β	β	β	β	c	c	$Y(\pi)$: *Labels*



Computing P(*s*, *y* | *M*)

$$P(s, y \mid M) = \sum_{\pi \mid Y(\pi)=y} P(s, \pi \mid M)$$

Only the path whose labelling is *y* have to be considered in the sum
In Forward and Backward algorithms it means to set

$$F_k(i) = 0, \;\; B_k(i) = 0 \quad \text{if} \quad Y(k) \neq y^i$$

		S	A	L	K	M	N	Y	T	R	E	I	M	V	A	S	N	Q	*s*: *Sequence*
		c	α	α	α	α	α	α	c	c	c	c	β	β	β	β	c	c	*y*: *Labels*
α_1	α																		
α_2	α																		
α_3	α																		
α_4	α																		
β_1	β																		
β_2	β																		
c	c																		

States Labelling

Baum-Welch training algorithm for labelled HMMs

Given a set of known labelled sequences (e.g. amino acid sequences and their native secondary structure) we want to find the parameters of the model, without knowing the generating paths:

$$\theta^{ML} = \text{argmax}_{\theta} \, [P \, (\, s, y \mid \theta, M)]$$

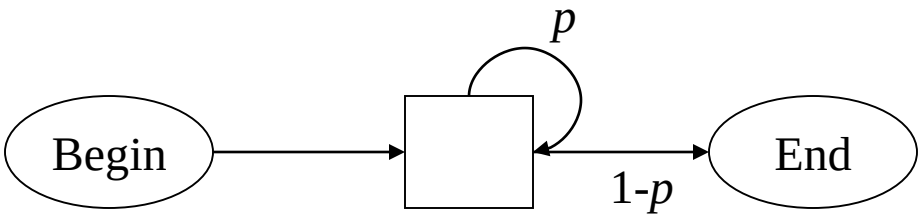
The algorithm is the same as in the non-labelled case if we use the Forward and Backward matrices defined in the last slide.

Supervised learning of the mapping

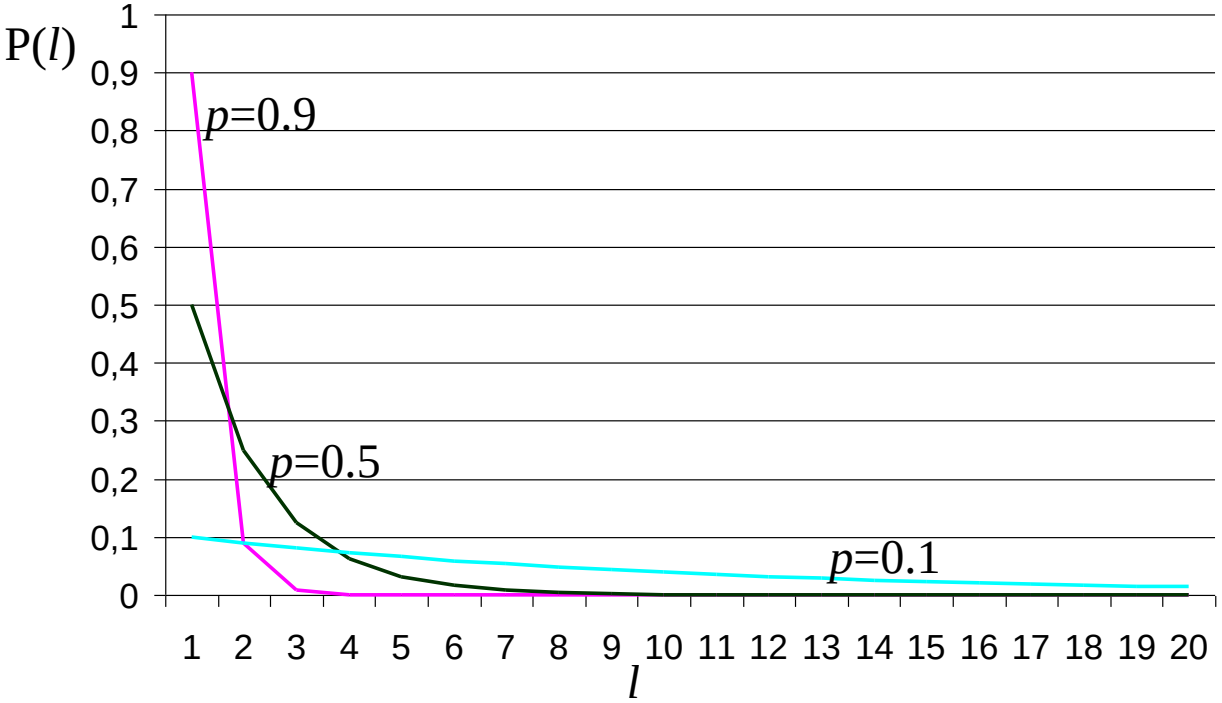
HMMs for Mapping problems

- Mapping problems in protein prediction
- Labelled HMMs
- Duration modelling

Self loops and geometric decay



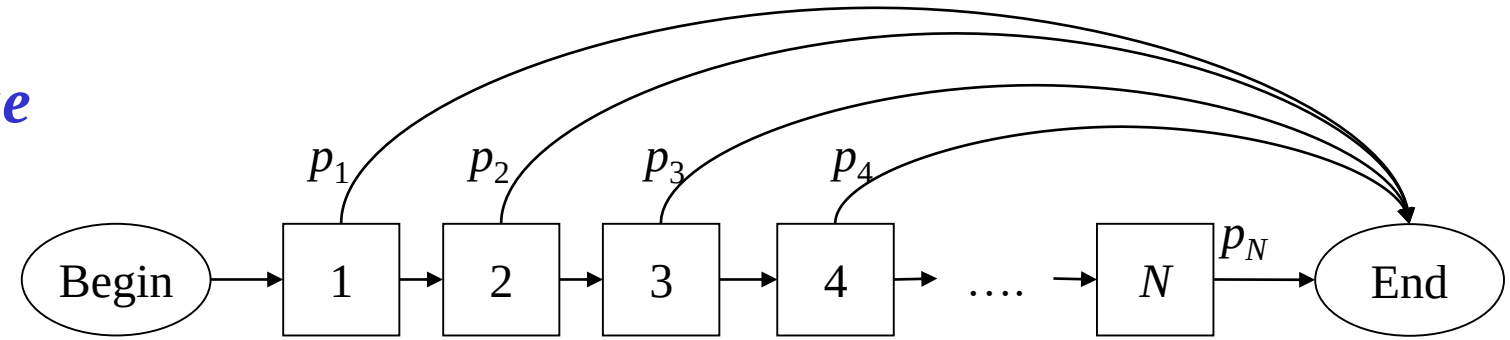
$$P(l) = p^{\,l-1} \cdot (\,1-p\,)$$

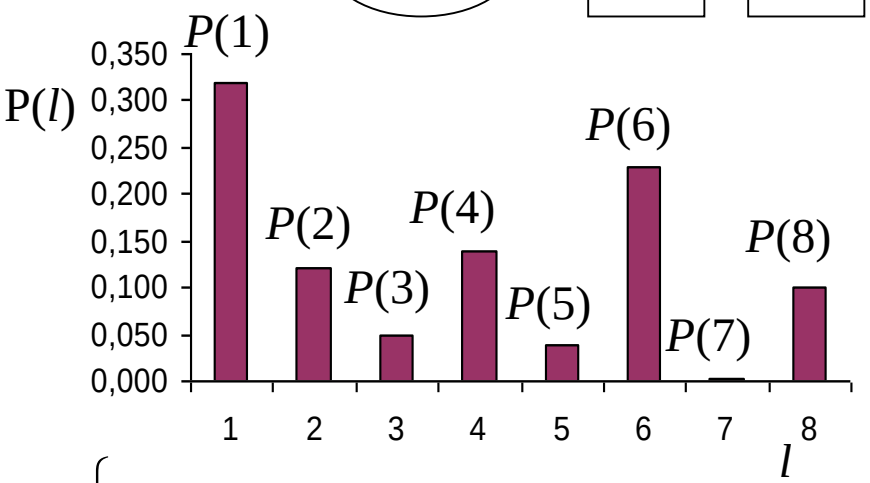


The length distribution of the generated segments is always exp-like

How can we model other length distributions?

Limited case



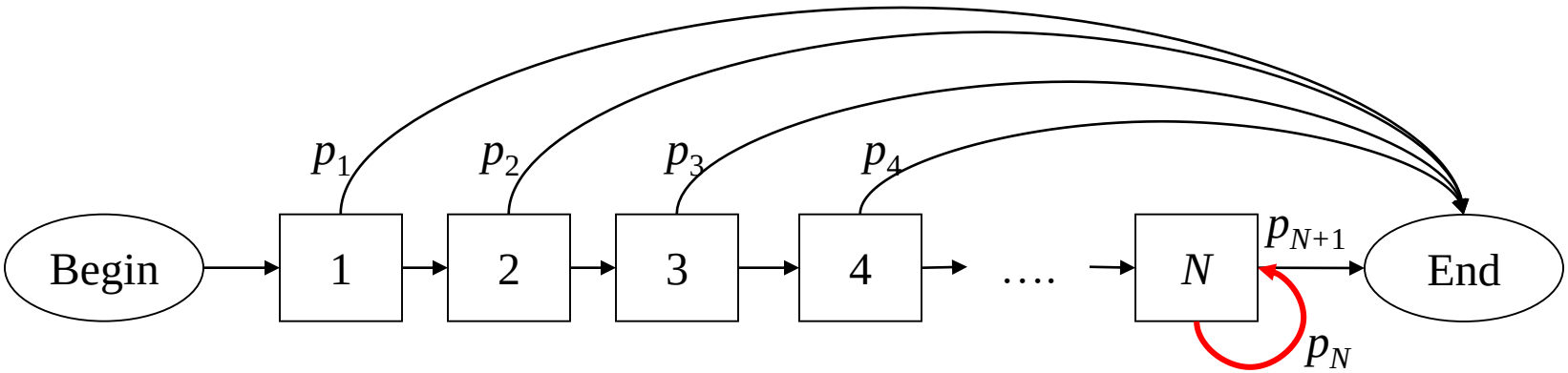


This topology can model any length distribution between 1 and N

$$\begin{cases} P(1) = p_1 \\ P(2) = (1-p_1)\cdot p_2 \\ P(3) = (1-p_1)\cdot(1-p_2)\cdot p_3 \\ \dots\dots\dots\dots\dots\dots \\ P(N) = \sum_{i=1}^{N-1}(1-p_i)\cdot p_N \end{cases} \Rightarrow \begin{cases} p_1 = P(1) \\ p_2 = P(2)/(1-p_1) \\ \dots\dots\dots\dots \\ p_k = \dfrac{P(k)}{\sum_{i=1}^{k-1}(1-p_i)} \end{cases}$$

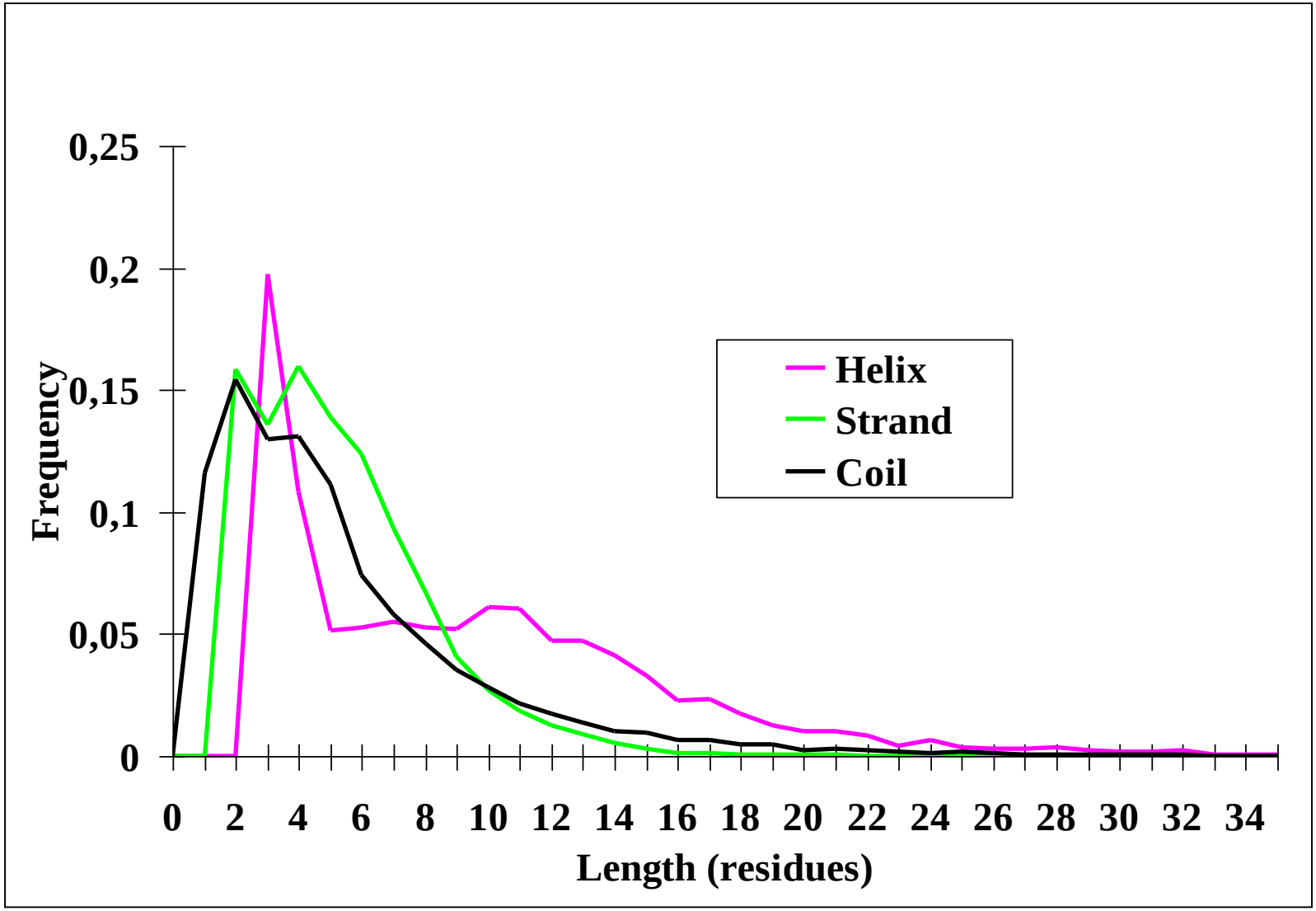
How can we model other length distributions?

Non limited case

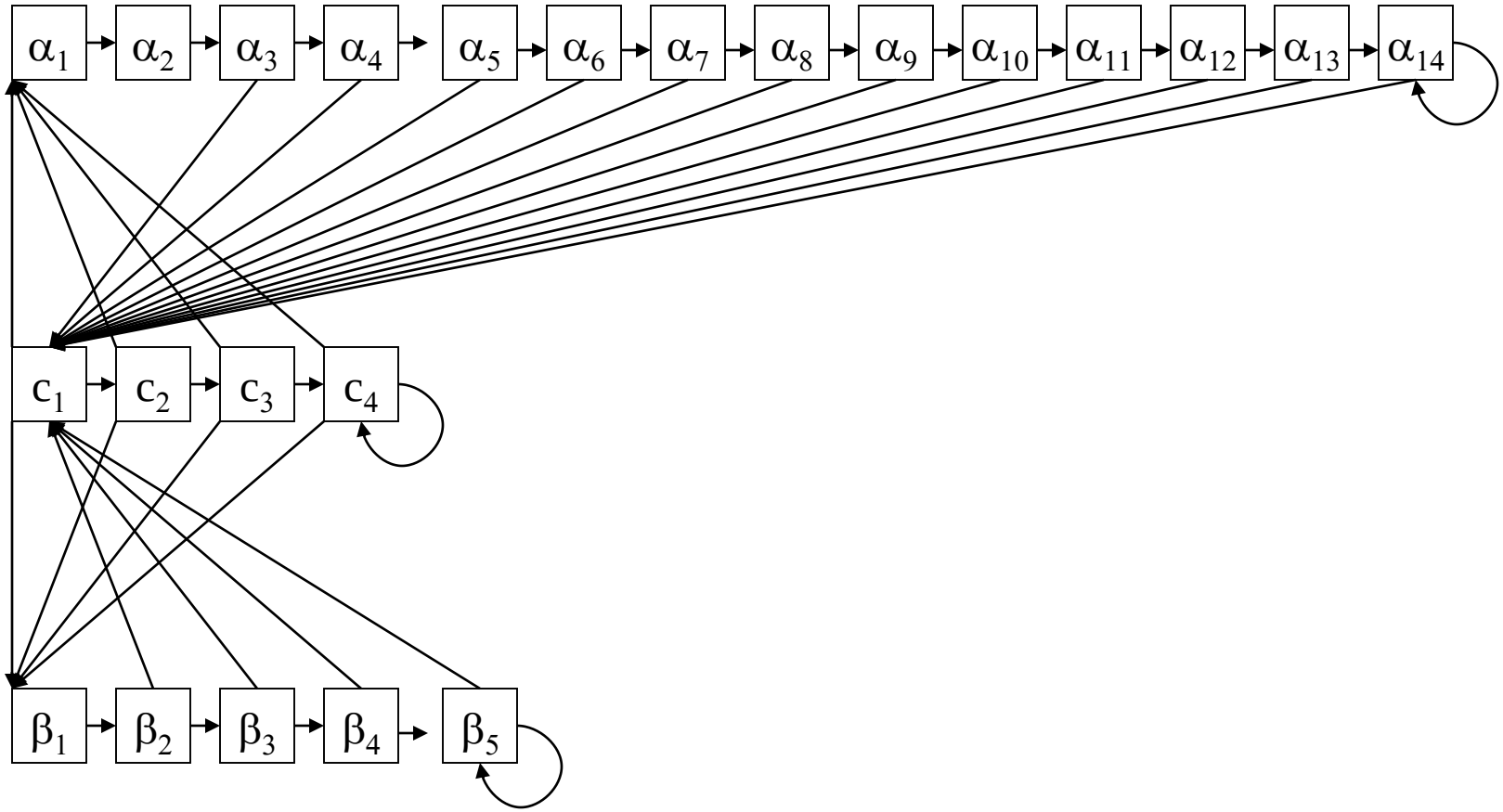


This topology can model any length distribution between 1 and N-1 and a geometrical decay from N and ∞

Secondary structure: length statistic



Secondary structure: model



Do we use the same emission probabilities for states sharing the same label?

HMMs for Mapping problems

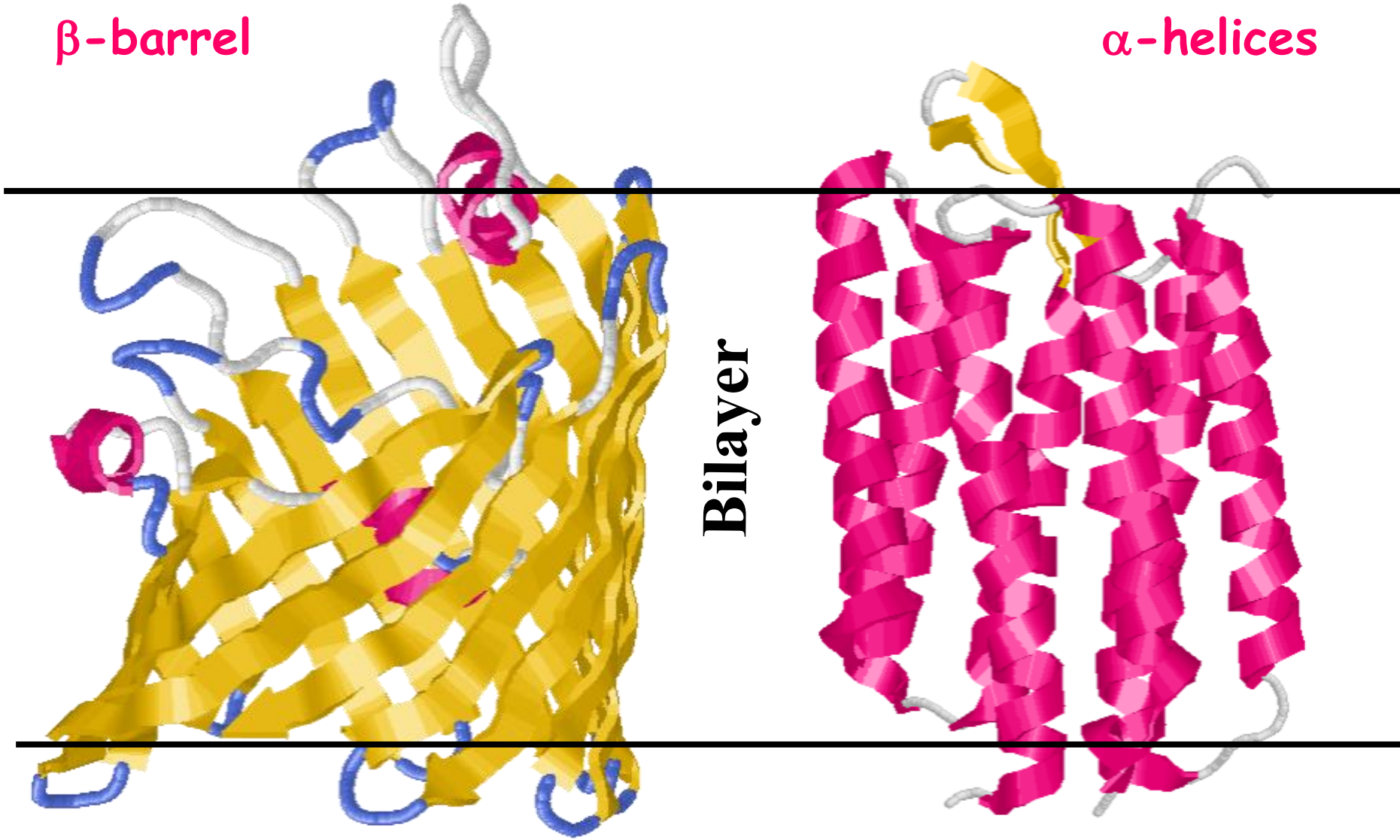
- Mapping problems in protein prediction
- Labelled HMMs
- Duration modelling
- Models for membrane proteins

Outer Membrane

β-barrel

Inner Membrane

α-helices



Porin
(*Rhodobacter capsulatus*)

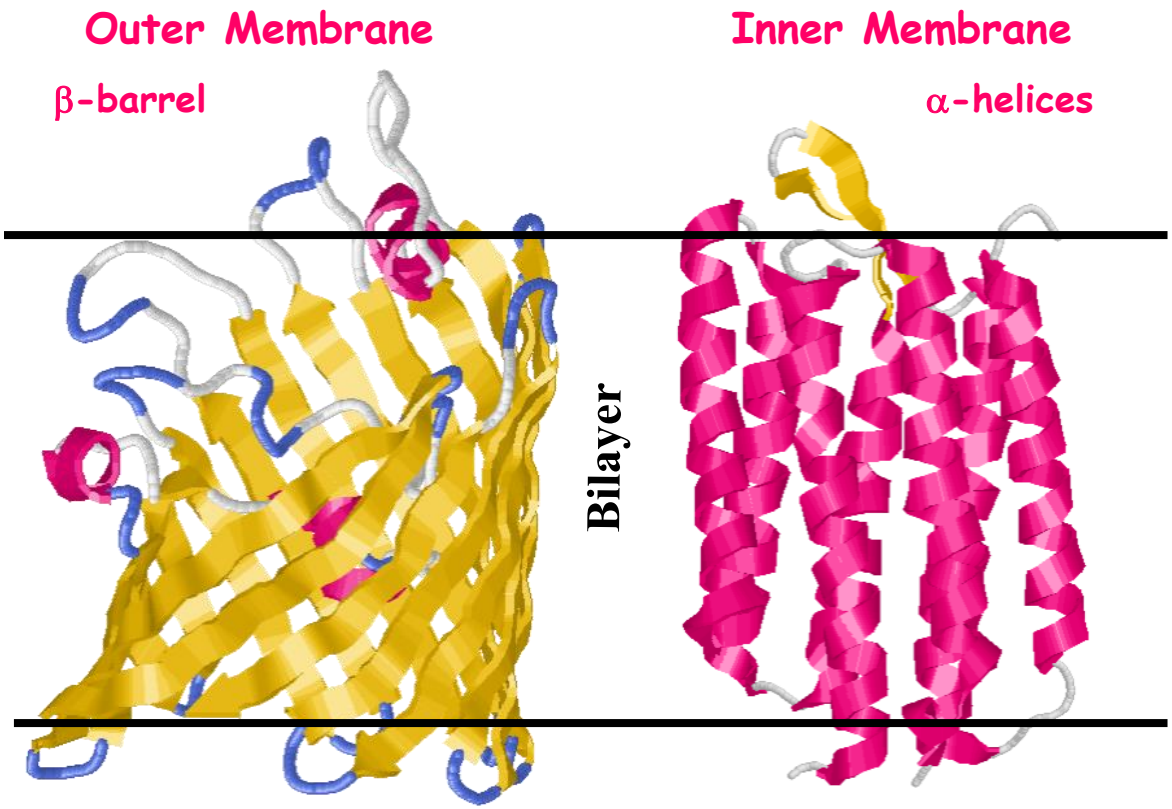
Bacteriorhodopsin
(*Halobacterium salinarum*)

Topology of membrane proteins

Topography

position of Trans Membrane Segments along the sequence

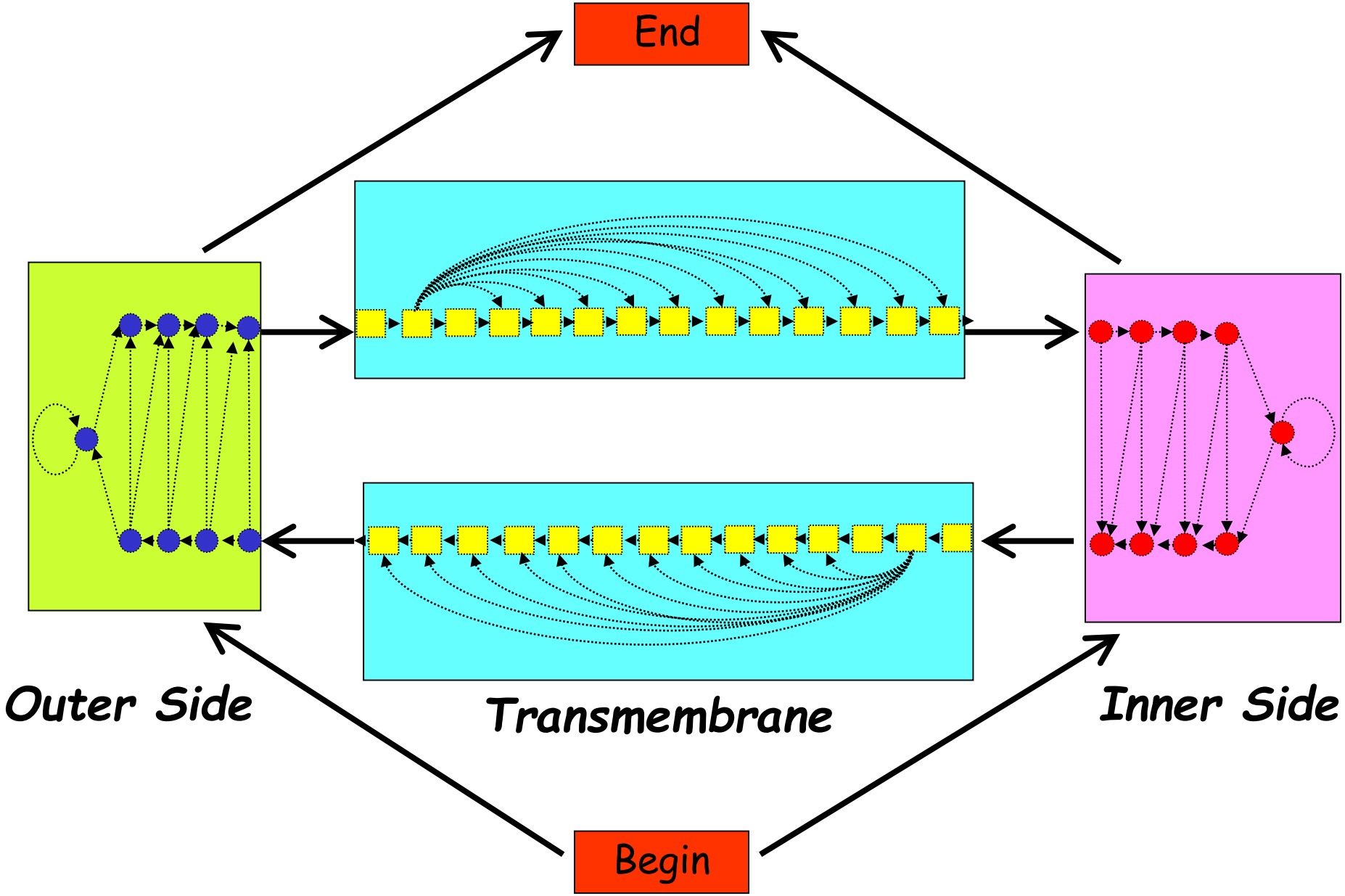
ALALMLCMLTYRHKELKLKLKK ALALMLCMLTYRHKELKLKLKK ALALMLCMLTYRHKELKLKLKK



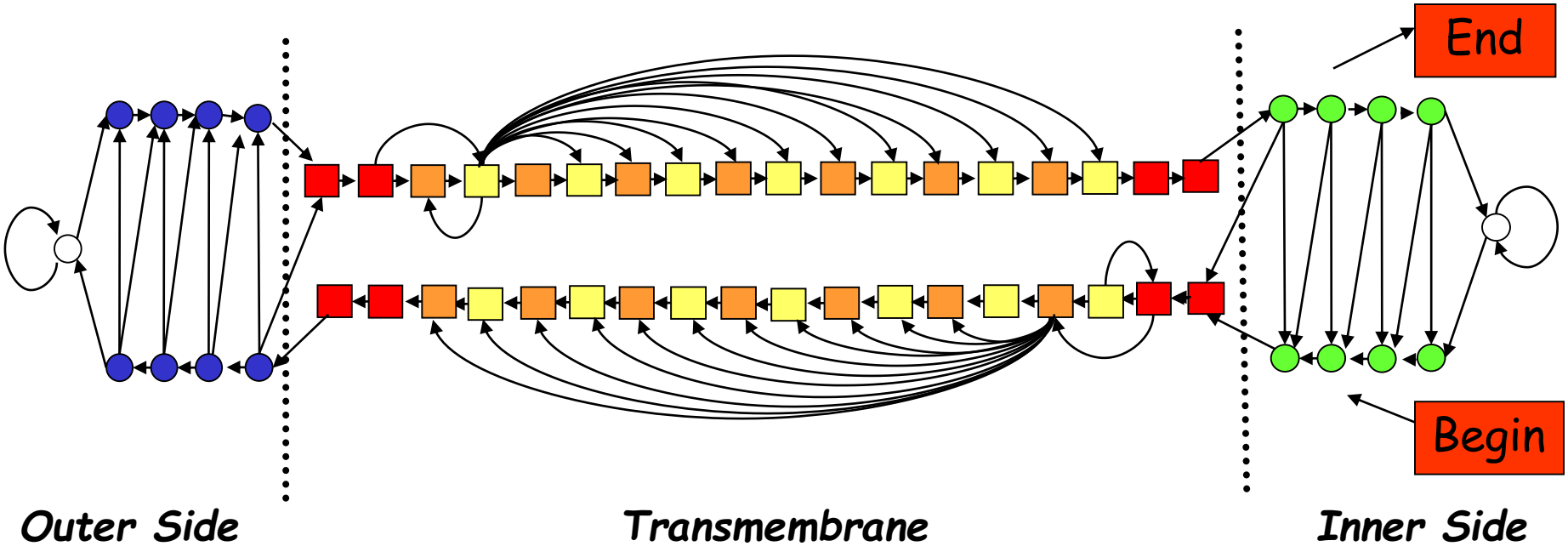
Porin
(*Rhodobacter capsulatus*)

Bacteriorhodopsin
(*Halobacterium salinarum*)

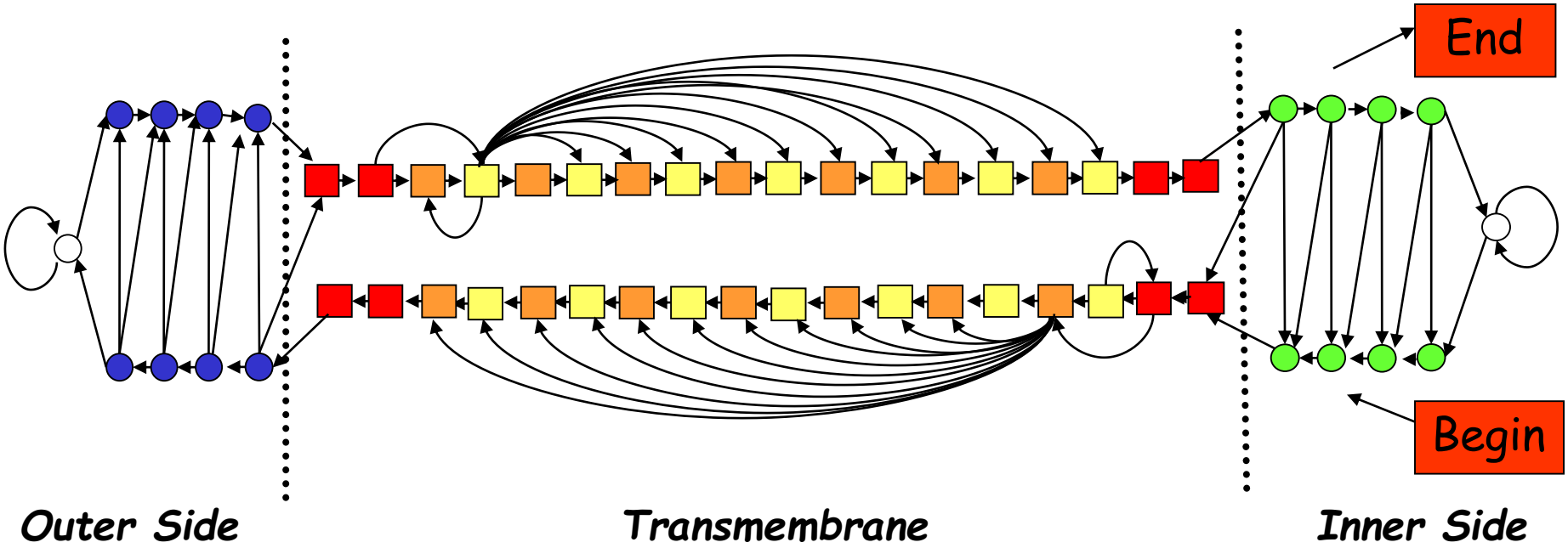
A generic model for membrane proteins (TMHMM)



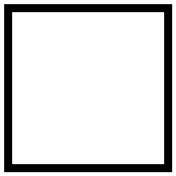
Model of β-barrel membrane proteins



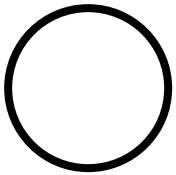
Model of β-barrel membrane proteins



Labels:

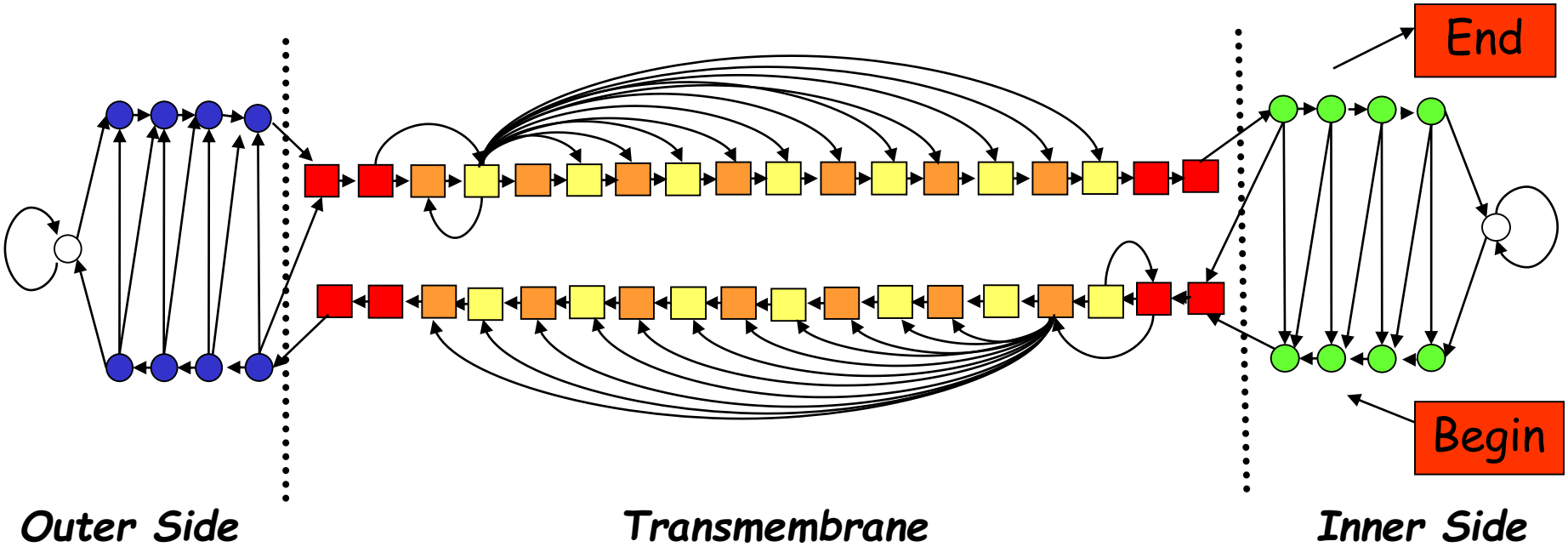


Transmembrane states



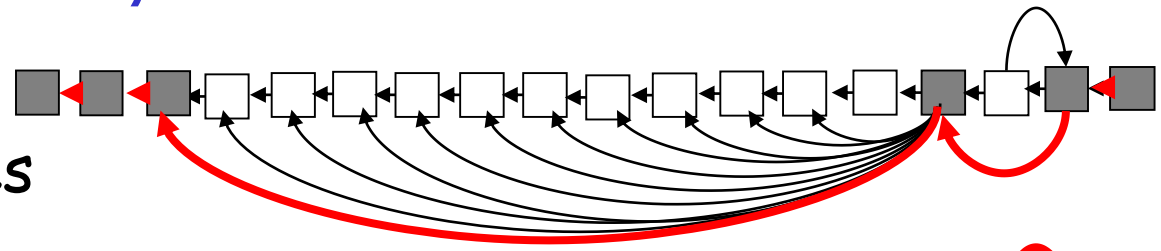
Loop states

Model of β-barrel membrane proteins

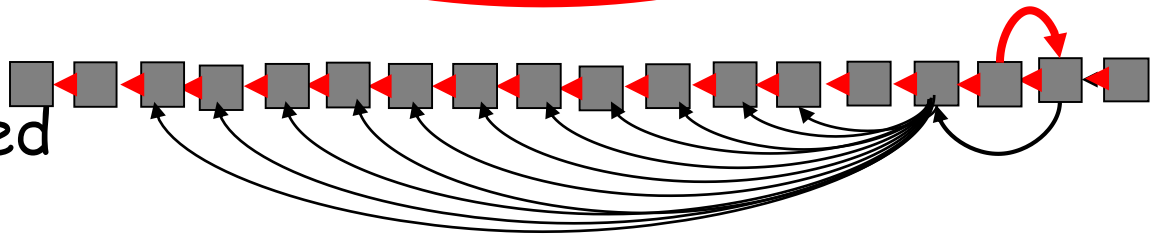


Length of transmembrane β-strands:

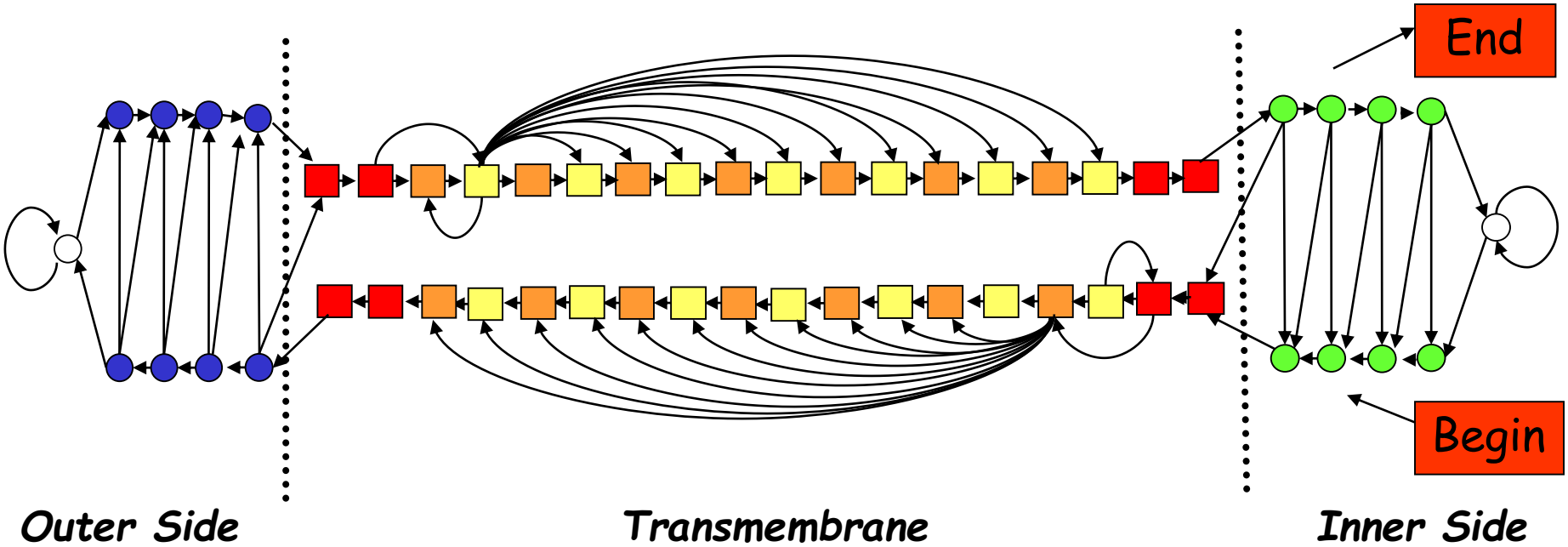
Minimum: 6 residues



Maximum: unbounded



Model of β-barrel membrane proteins



Six different sets of emission parameters:

Outer loop

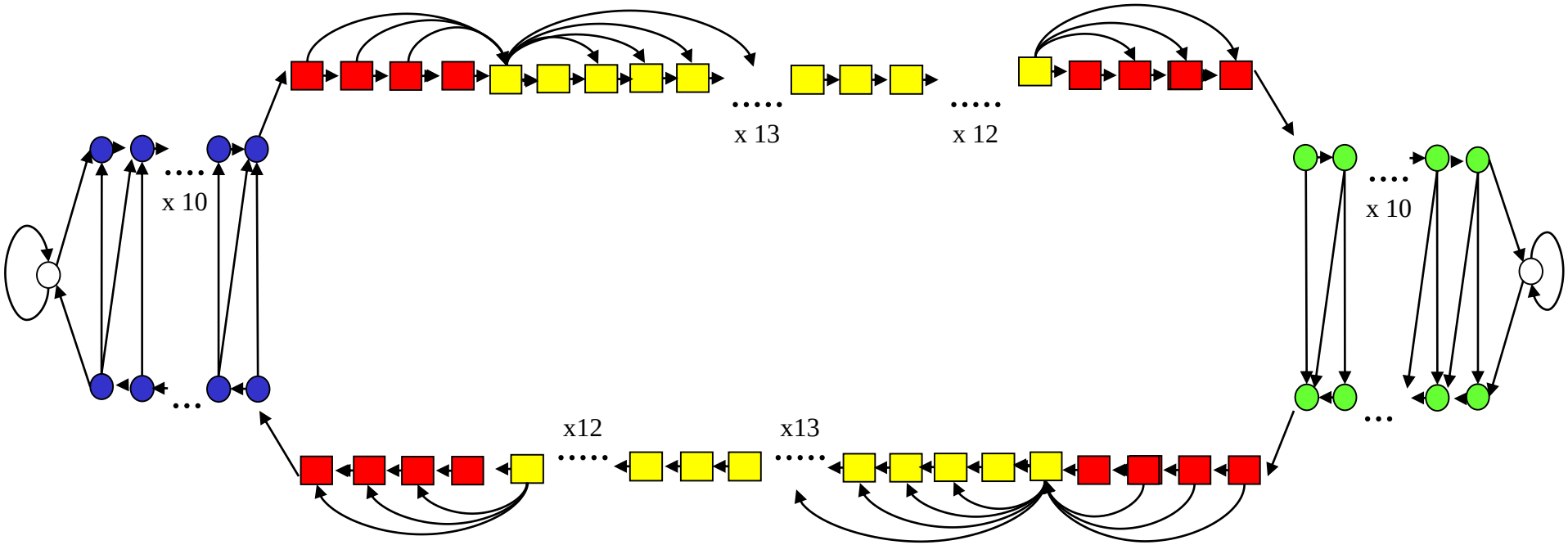
Inner loop

Long globular domains

TM strands edges

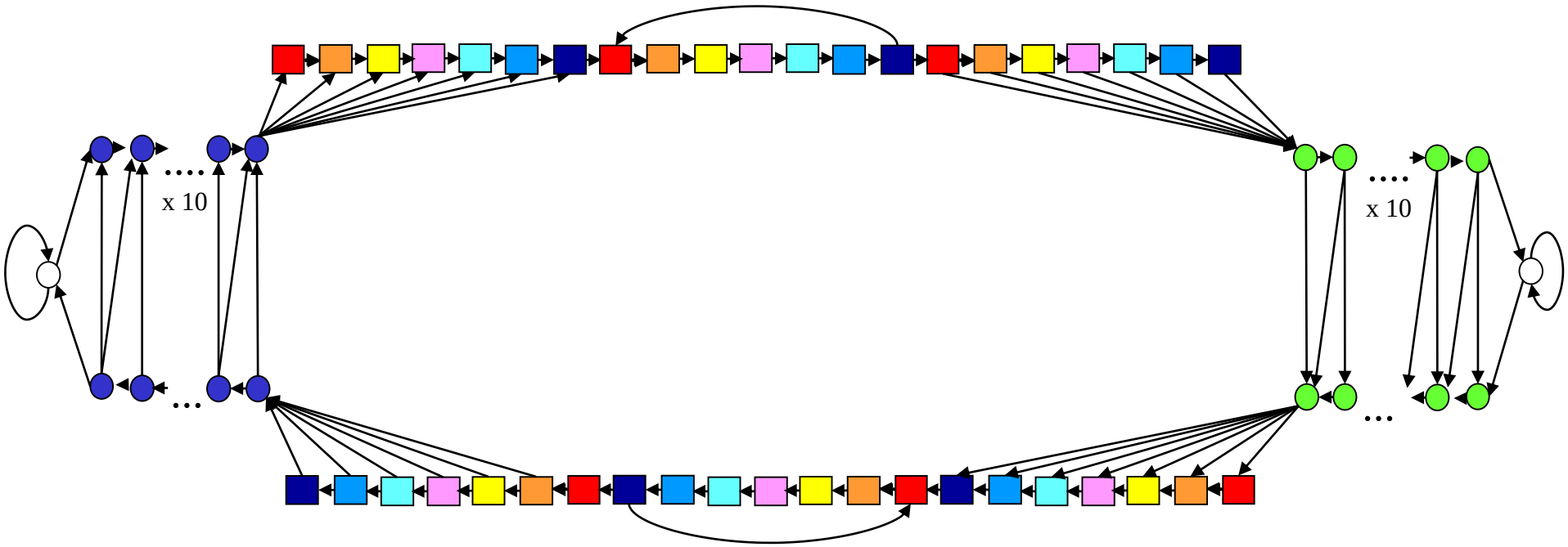
TM strands core

Model of α-helix membrane proteins (HMM1)

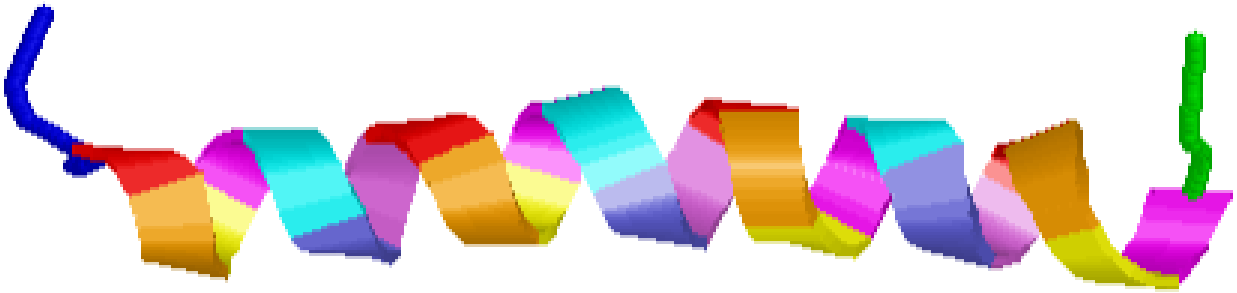


Outer Side ***Transmembrane*** ***Inner Side***

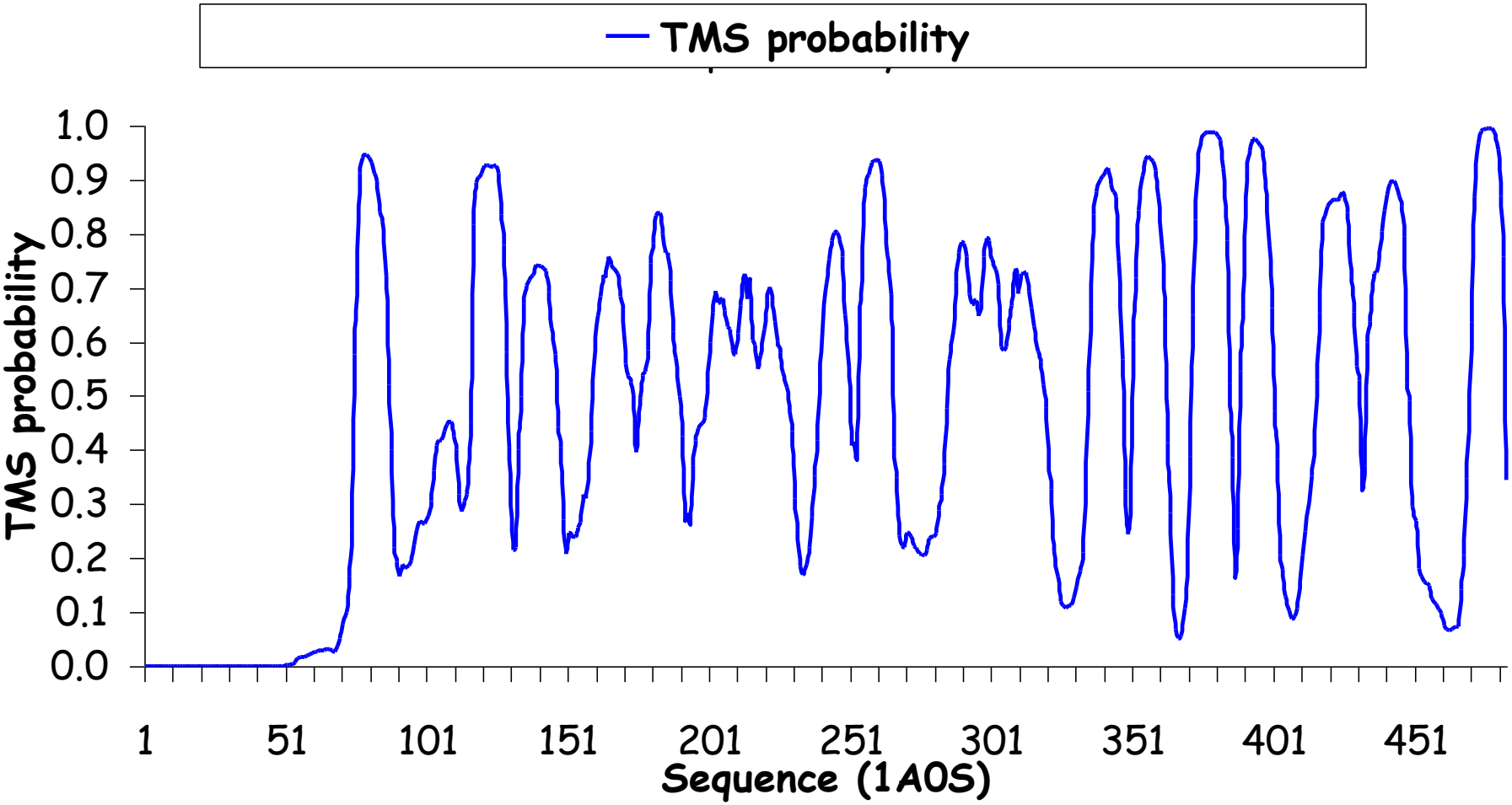
Model of α-helix membrane proteins (HMM2)



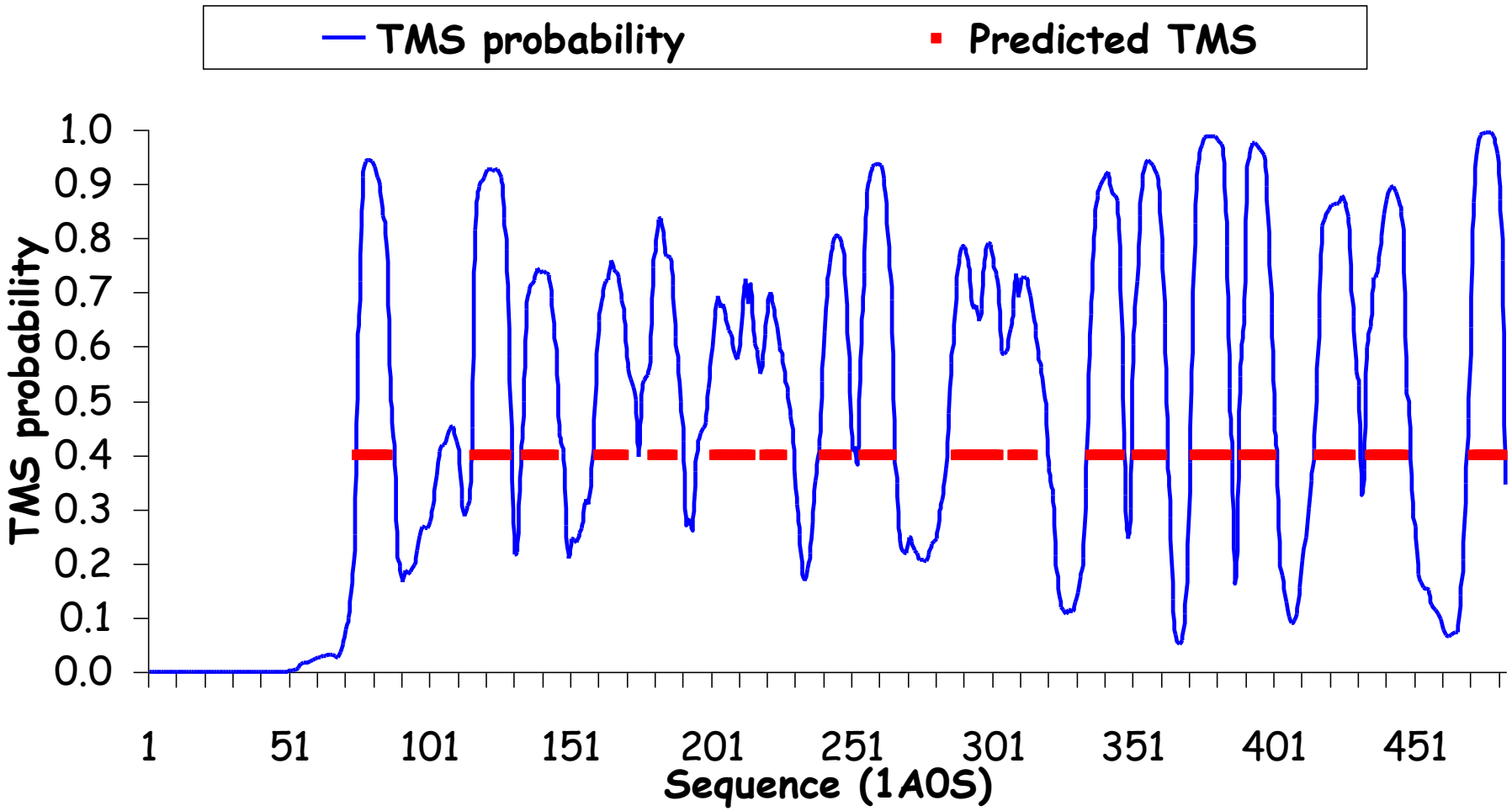
Outer Side ***Transmembrane*** ***Inner Side***



Dynamic programming filtering procedure

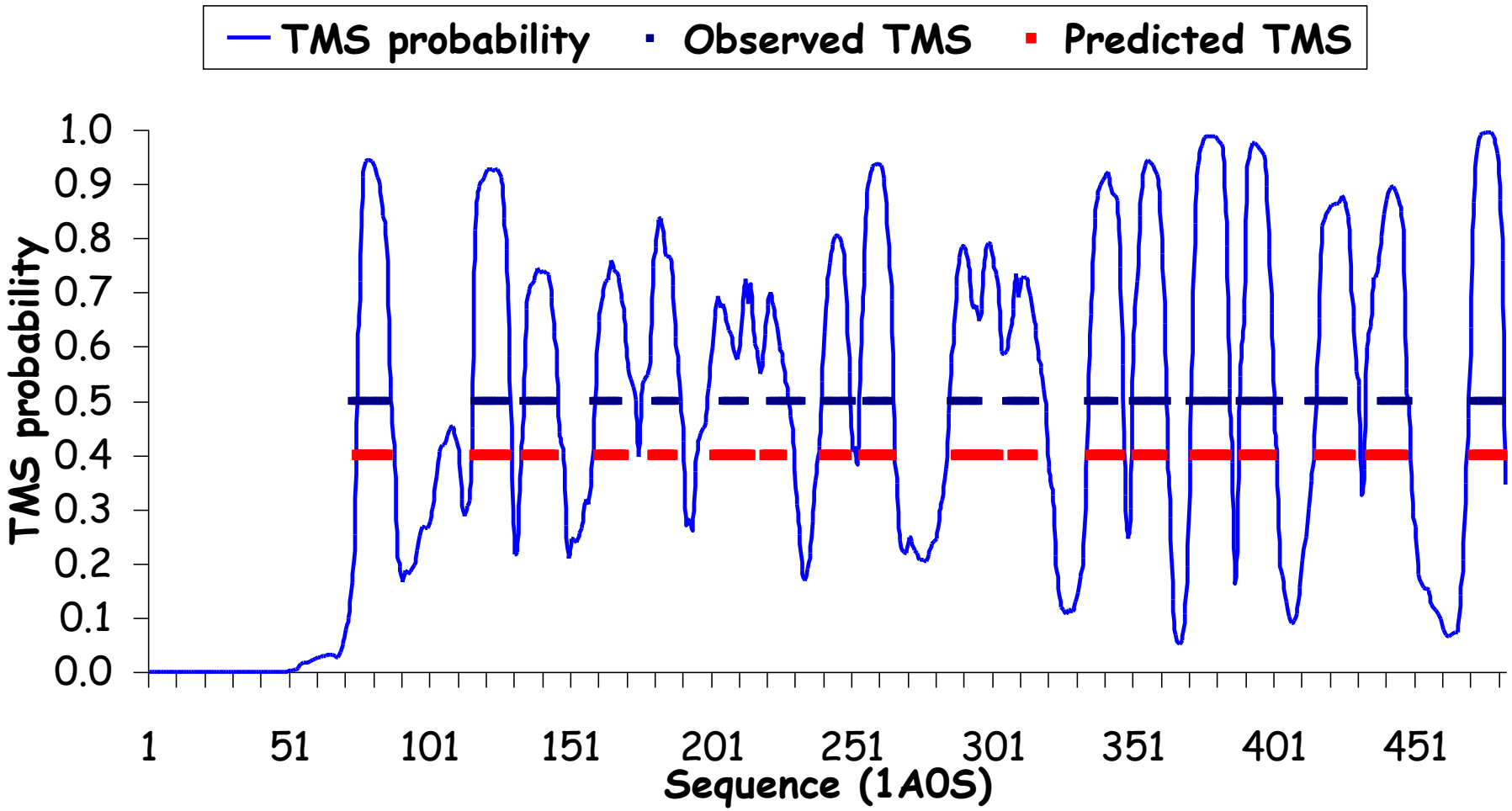


Dynamic programming filtering procedure



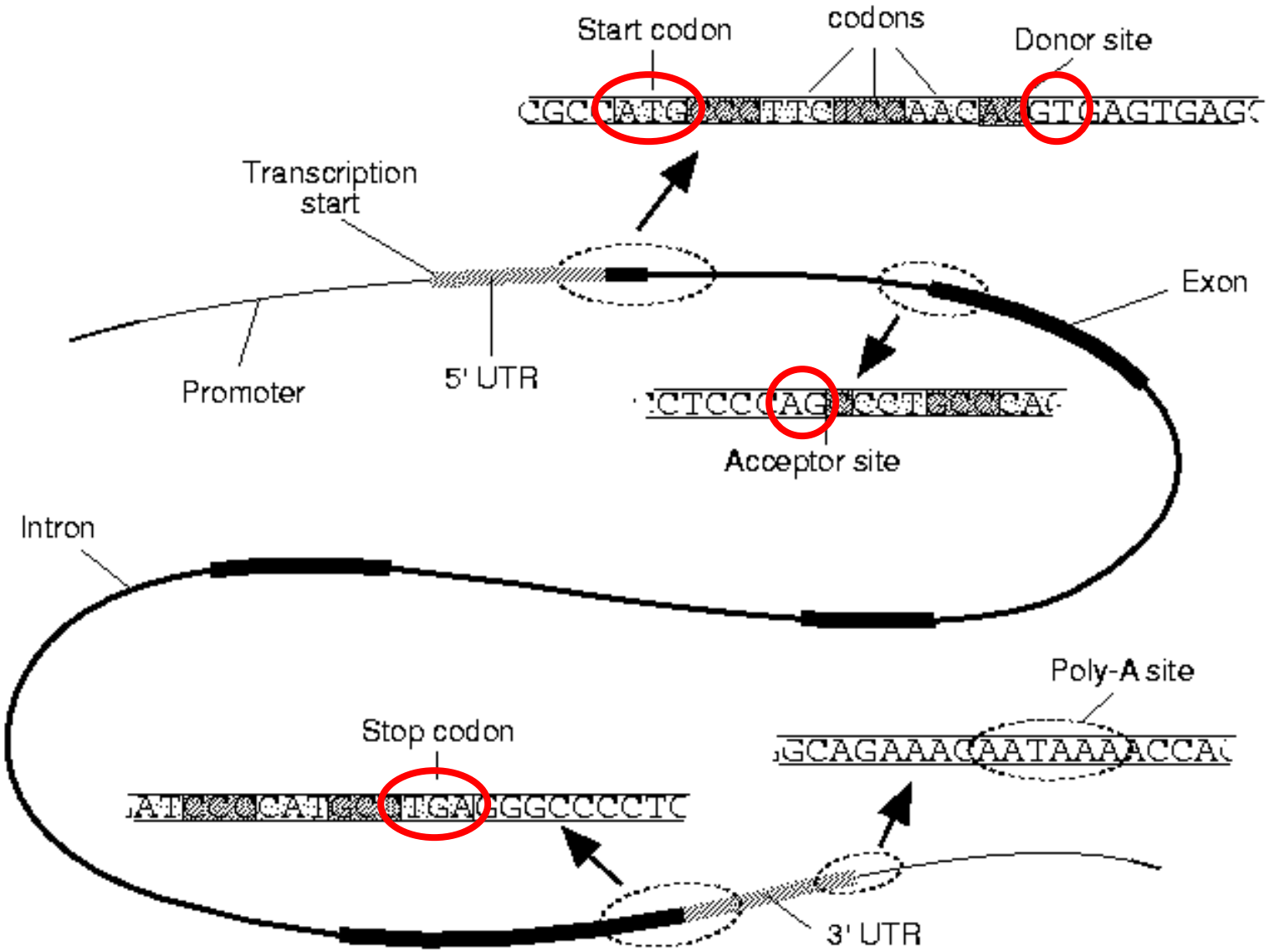
Maximum-scoring subsequences with constrained segment length and number

Dynamic programming filtering procedure



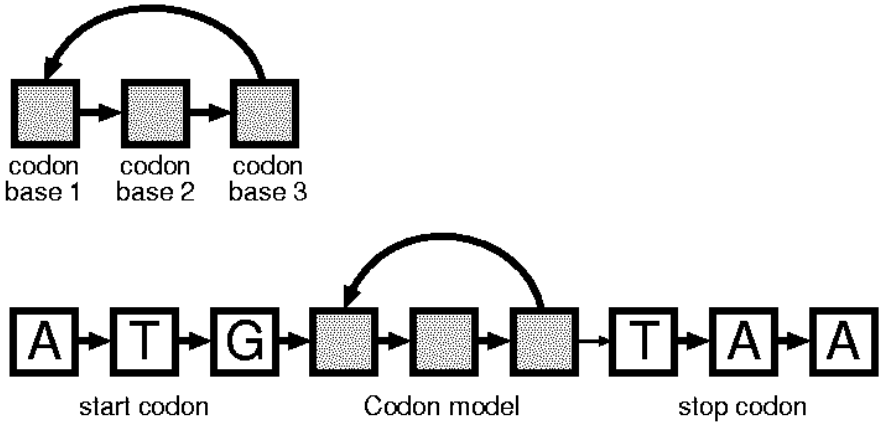
Maximum-scoring subsequences with constrained segment length and number

Eukaryotic gene structure



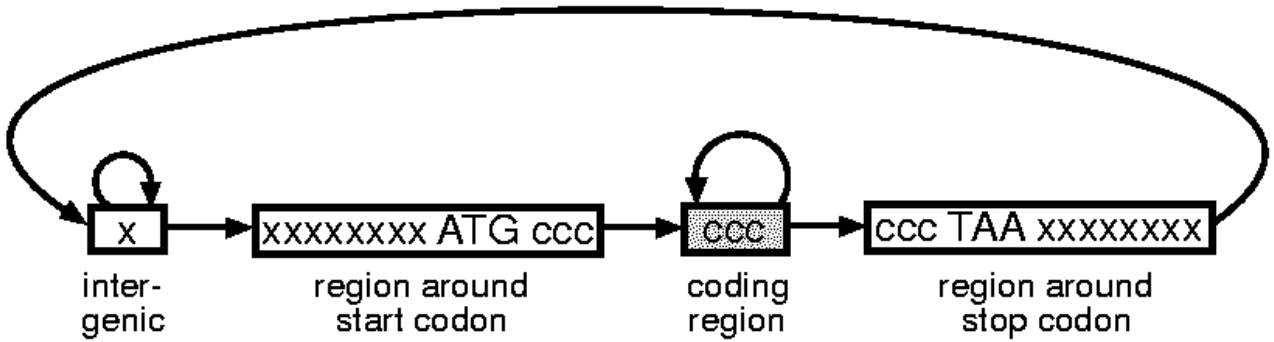
A. Krogh

Simple model for coding regions



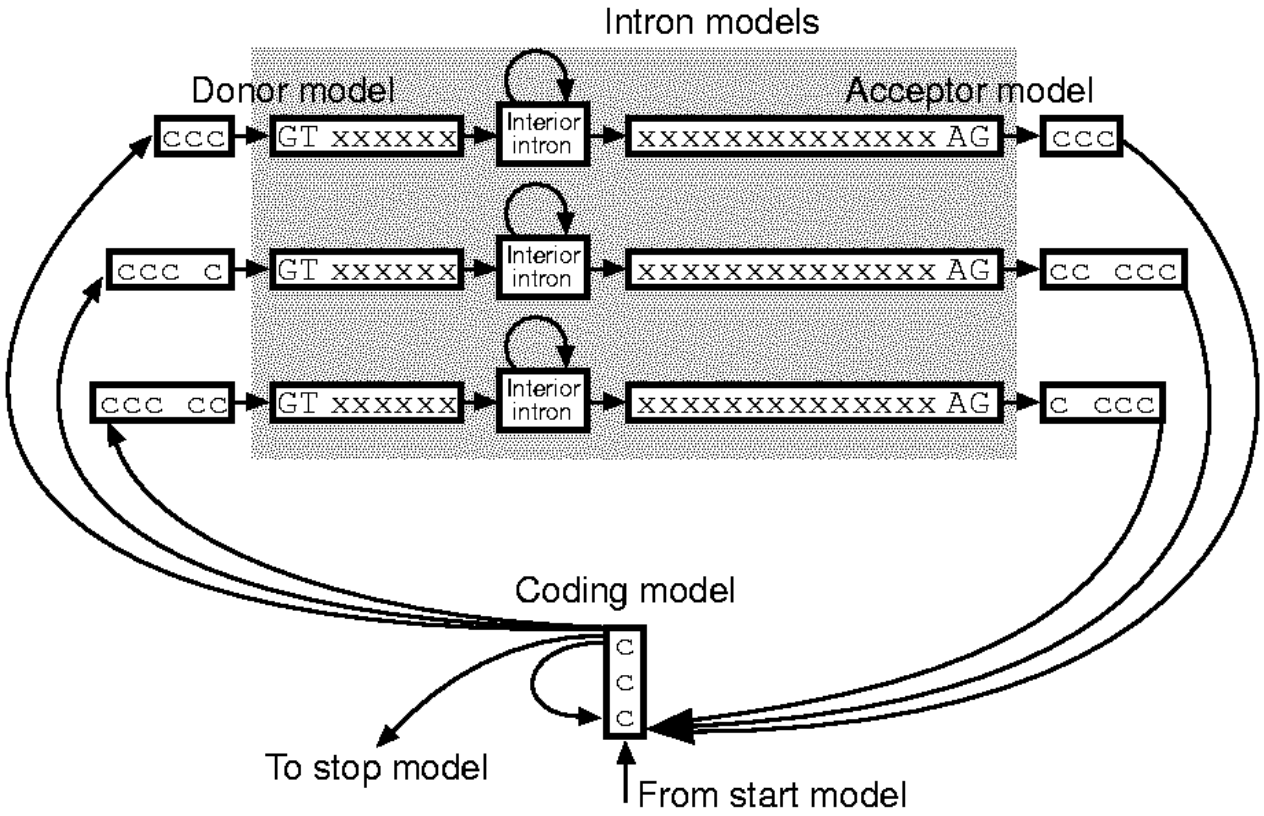
A. Krogh

Simple model for unspliced gene



A. Krogh

Simple model for spliced gene



A. Krogh