Hidden Markov models

Wessel van Wieringen
w.n.van.wieringen@vu.nl

Department of Epidemiology and Biostatistics, VUmc
& Department of Mathematics, VU University
Amsterdam, The Netherlands

vrije Universiteit

VU medisch centrum

Hidden Markov model

Consider (temporarily) a binary DNA sequence:

...

010101010100101010100100100010101001100

101010101111111111111111111111111111111

111111111111111111111111111110101010100

101010100101001010101001010101111111111

111111111111111111111111111111111111111

111111111111101010101010101010101001010

101010101010101010101010101001010100010

100110110101111111111111111111111111111

...

Question: anything striking?

Hidden Markov model

Consider (temporarily) a binary DNA sequence:

```
...
010101010100101010100100100010101001100
101010101111111111111111111111111111111
111111111111111111111111111110101010100
101010100101001010101001010101111111111
111111111111111111111111111111111111111
111111111111101010101010101010101001010
101010101010101010101010101001010100010
100110110101111111111111111111111111111
...
```

Question: can the sequence be modeled by a Markov chain?

Hidden Markov model

You would need two Markov chains to do this, e.g.:

- one with a transition matrix like

$$\mathbf{P} = \begin{pmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{pmatrix}$$

to generate something like

01010101010010101010010010001010100 1100

- and one with a transition matrix like

$$\mathbf{P} = \begin{pmatrix} 0.0 & 1.0 \\ 0.0 & 1.0 \end{pmatrix}$$

to generate something like

11111111111111111111111111111111111111

Hidden Markov model

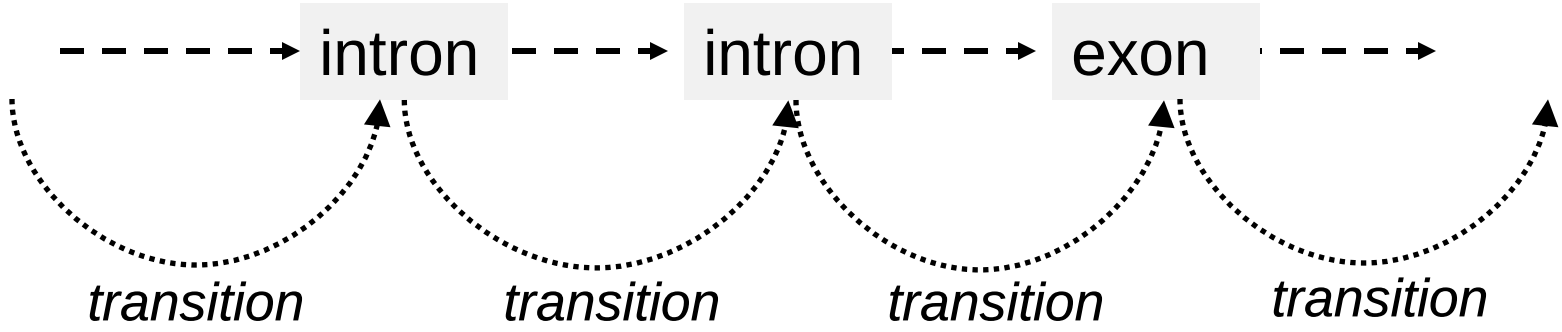
Consider a charicature of the DNA that consists of introns and exons only.

Introns and exons are characterized by other nucleotide distributions.

In addition, assume that start and end base pair positions of introns and exons are unknown.

Hidden Markov model

What does this look like?

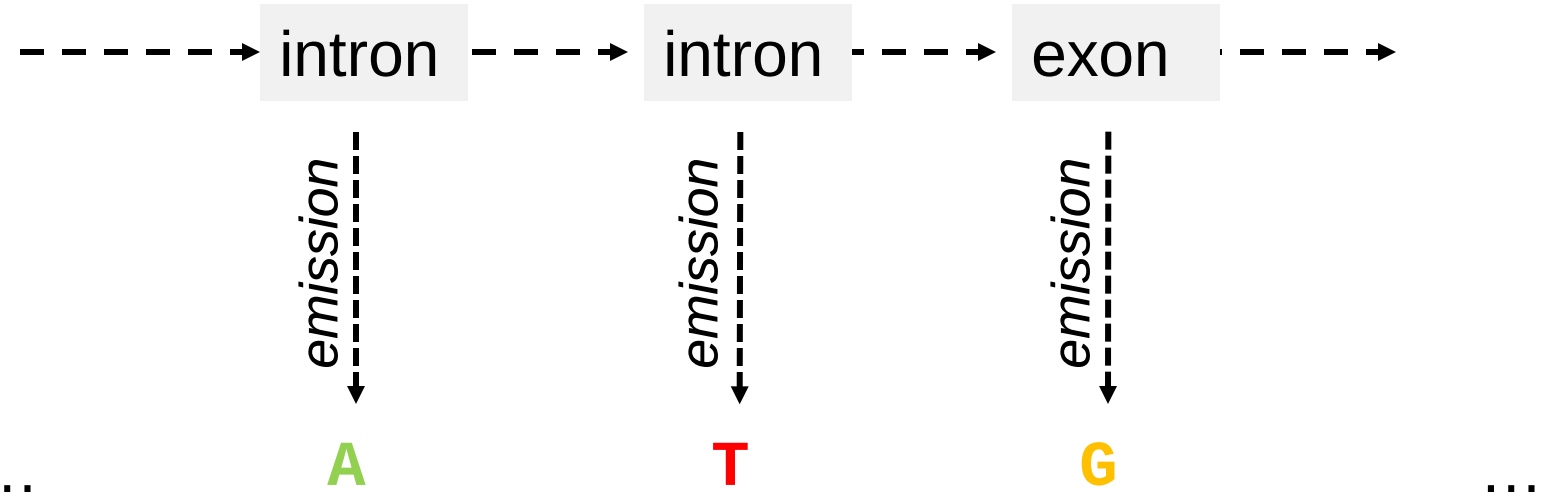


This sequence:

- is unobserved.
- may be modelled by a 1st order Markov chain.

Hidden Markov model

How to obtain the observed sequence of nucleotides?

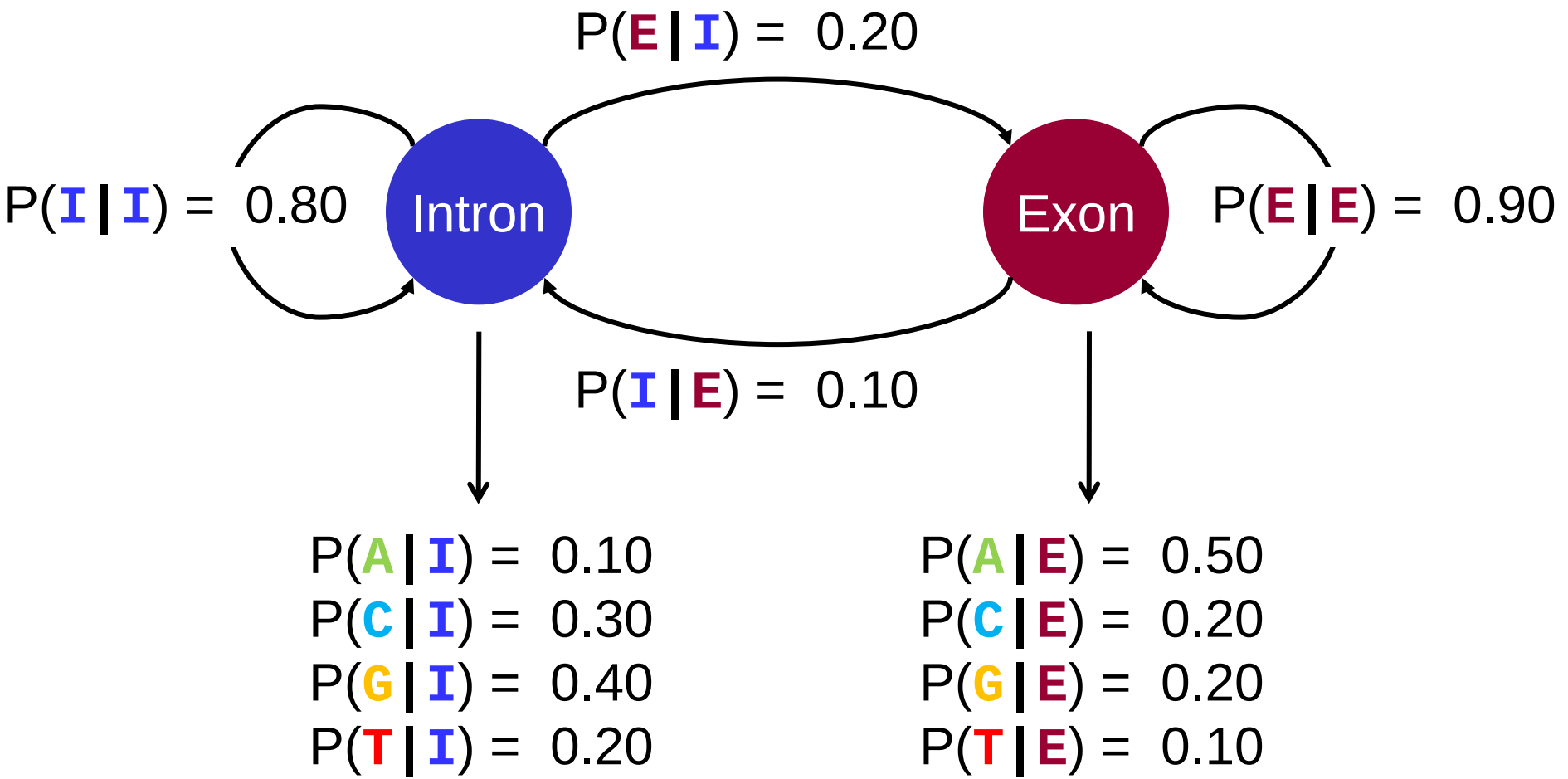


Given the state (intron / exon):

- simply sample from (say) a multinomial.

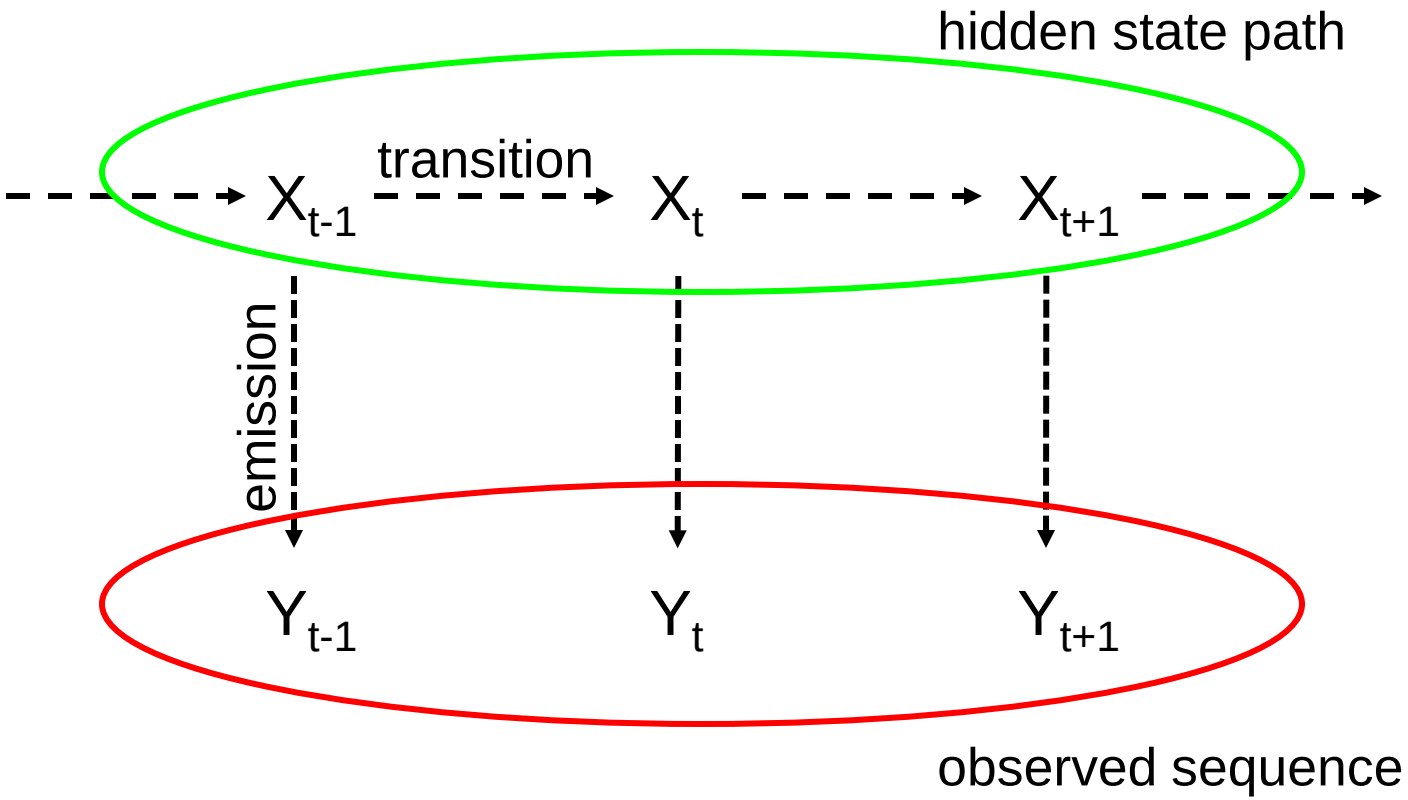
Hidden Markov model

Intron-exon example



Hidden Markov model

Architecture of a hidden Markov model



book Axelson-Fisk (2010): Section 2.1.2

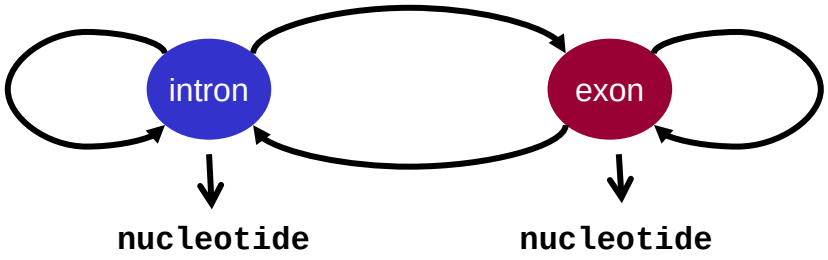
Hidden Markov model

Definition of a HMM

The processes $\{X_t\}_{t=1,2,...}$ and $\{Y_t\}_{t=1,2,...}$ form a HMM if:

1) $\{X_t\}_{t=1,2,...}$ is a discrete, time-homogeneous Markov chain with state space $S = \{E_1, ..., E_S\}$, transition matrix $\mathbf{P} = (p_{ij})$ and initial distribution $\boldsymbol{\pi}=(\pi_1, ..., \pi_S)^T$.

In the intron-exon example:



$$\mathbf{P} = \begin{array}{c} \\ \mathbf{I} \\ \mathbf{E} \end{array} \begin{array}{c} \begin{array}{cc} \mathbf{I} & \mathbf{E} \end{array} \\ \begin{pmatrix} 0.80 & 0.20 \\ 0.10 & 0.90 \end{pmatrix} \end{array} \quad \text{with} \quad \boldsymbol{\pi} = \begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix} \begin{array}{l} \mathbf{I} \\ \mathbf{E} \end{array}$$

book Axelson-Fisk (2010): Section 2.1.2

Hidden Markov model

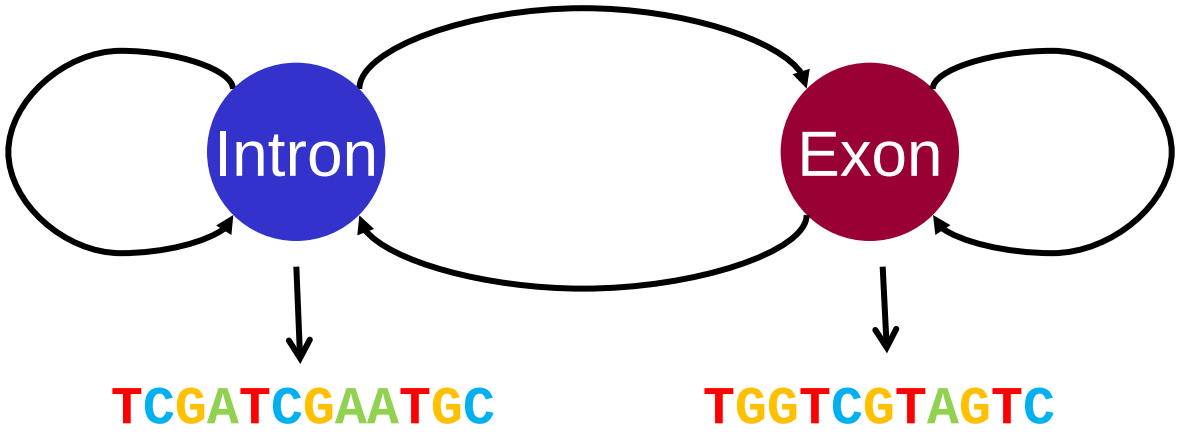
Definition of a HMM (continued)

The processes $\{X_t\}_{t=1,2,...}$ and $\{Y_t\}_{t=1,2,...}$ form a HMM if:

2) $\{Y_t\}_{t=1,2,...}$ is an observable stochastic process with state space $V = \{V_1,...,V_m\}$.

In the intron-exon example:

The DNA sequence is observable and V = {**A**, **C**, **G**, **T**}



book Axelson-Fisk (2010): Section 2.1.2

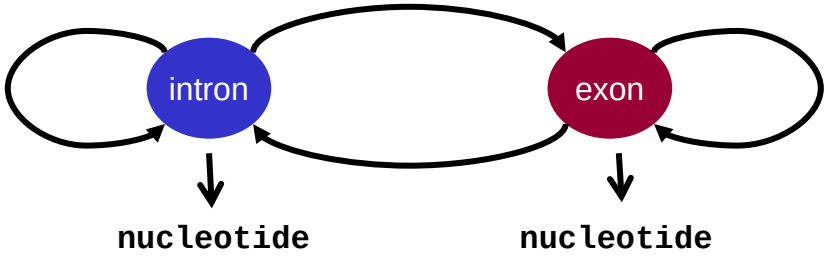
Hidden Markov model

Definition of a HMM (continued)

The processes $\{X_t\}_{t=1,2,...}$ and $\{Y_t\}_{t=1,2,...}$ form a HMM if:

3) $\{X_t\}_{t=1,2,...}$ and $\{Y_t\}_{t=1,2,...}$ are related through the conditional probabilities $P(Y_t=V_j \mid X_t = E_i) = b_i(V_j) = b_{ij}$. The matrix $\mathbf{B} = (b_{ij})$ is called the *emission matrix*.

In the intron-exon example:



$$\mathbf{B} = \begin{array}{c} \\ \mathtt{I} \\ \mathtt{E} \end{array} \begin{array}{c} \begin{array}{cccc} \mathtt{A} & \mathtt{C} & \mathtt{G} & \mathtt{T} \end{array} \\ \begin{pmatrix} 0.10 & 0.30 & 0.40 & 0.20 \\ 0.50 & 0.20 & 0.20 & 0.10 \end{pmatrix} \end{array}$$

book Axelson-Fisk (2010): Section 2.1.2

Hidden Markov model

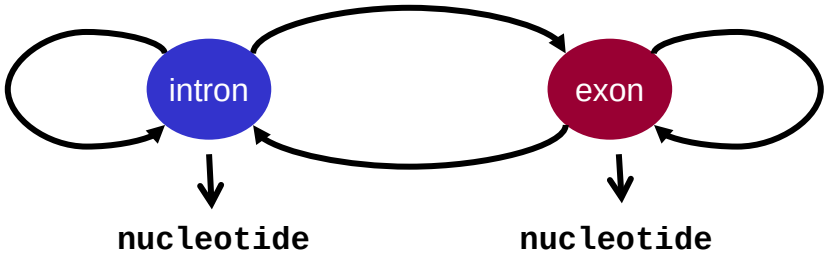
Definition of a HMM (continued)

The processes $\{X_t\}_{t=1,2,...}$ and $\{Y_t\}_{t=1,2,...}$ form a HMM if:

4) Given the states $\{X_t\}_{t=1,2,...}$ the observations $\{Y_t\}_{t=1,2,...}$ are independent.

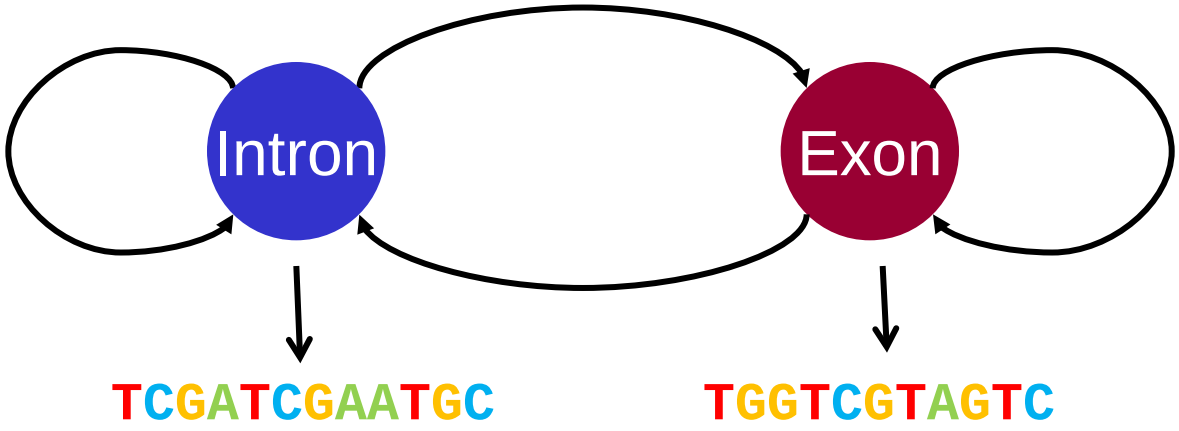
In the intron-exon example, e.g.:

$$P(Y_t=\mathbf{A}, Y_{t+1}=\mathbf{C} \mid X_t, X_{t+1}) =$$
$$P(Y_t=\mathbf{A} \mid X_t) * P(Y_{t+1}=\mathbf{C} \mid X_{t+1})$$



Hidden Markov model

Parametrization of the intron-exon sample



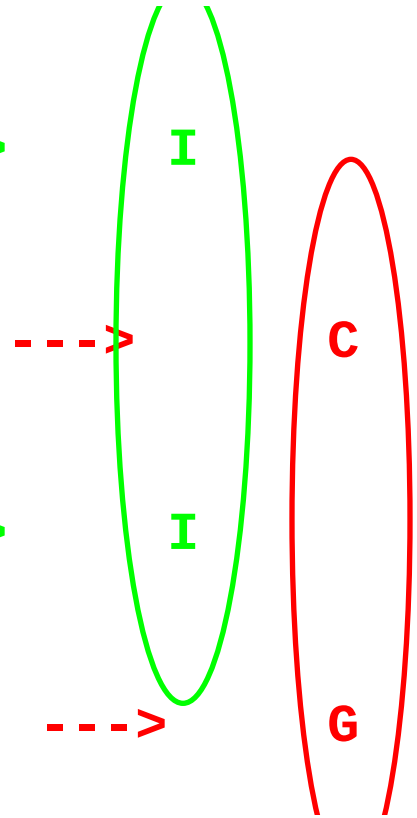
$$\mathbf{P} = \begin{array}{c} \\ \mathtt{I} \\ \mathtt{E} \end{array} \begin{array}{c} \begin{array}{cc} \mathtt{I} & \mathtt{E} \end{array} \\ \begin{pmatrix} 0.80 & 0.20 \\ 0.10 & 0.90 \end{pmatrix} \end{array} \quad \text{with} \quad \boldsymbol{\pi} = \begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix} \begin{array}{c} \mathtt{I} \\ \mathtt{E} \end{array}$$

$$\mathbf{B} = \begin{array}{c} \\ \mathtt{I} \\ \mathtt{E} \end{array} \begin{array}{c} \begin{array}{cccc} \mathtt{A} & \mathtt{C} & \mathtt{G} & \mathtt{T} \end{array} \\ \begin{pmatrix} 0.10 & 0.30 & 0.40 & 0.20 \\ 0.50 & 0.20 & 0.20 & 0.10 \end{pmatrix} \end{array}$$

Hidden Markov model

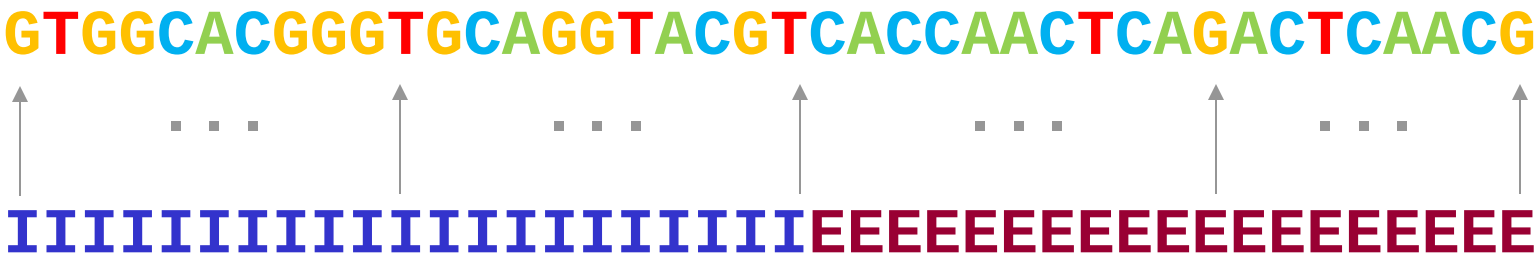
Sampling from this HMM:

```
t =  1 : P(I)=0.9, P(E)=0.1
                                        --->
       : P(A|I)=0.10, P(C|I)=0.30,
         P(G|I)=0.40, P(T|I)=0.20

t =  2 : P(I|I)=0.8, P(E|I)=0.2
                                        --->
       : P(A|I)=0.10, P(C|I)=0.30,
         P(G|I)=0.40, P(T|I)=0.20

...
```



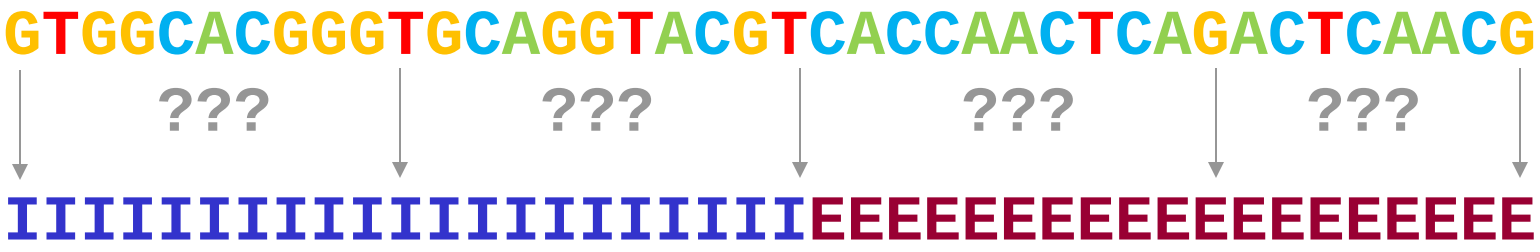
Hidden Markov model

A HMM may generate something like:



Practice

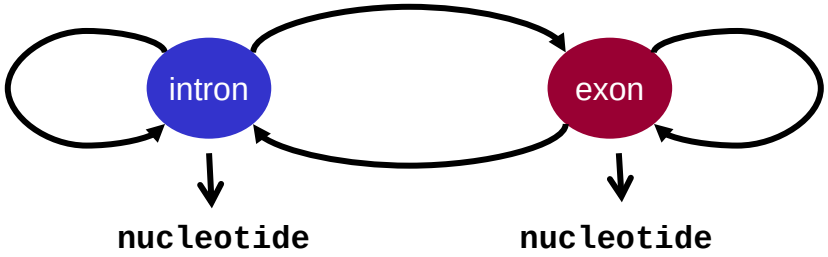
Can the intron-exon sequence be recovered from that of the nucleotides? I.e.:



Likelihood

Likelihood

Consider the following parametrization of the intron-exon example.



$$\mathbf{P} = \begin{array}{c} \\ \mathtt{I} \\ \mathtt{E} \end{array} \begin{array}{c} \begin{array}{cc} \mathtt{I} & \mathtt{E} \end{array} \\ \begin{pmatrix} 0.80 & 0.20 \\ 0.10 & 0.90 \end{pmatrix} \end{array} \quad \text{with} \quad \boldsymbol{\pi} = \begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix} \begin{array}{l} \mathtt{I} \\ \mathtt{E} \end{array}$$

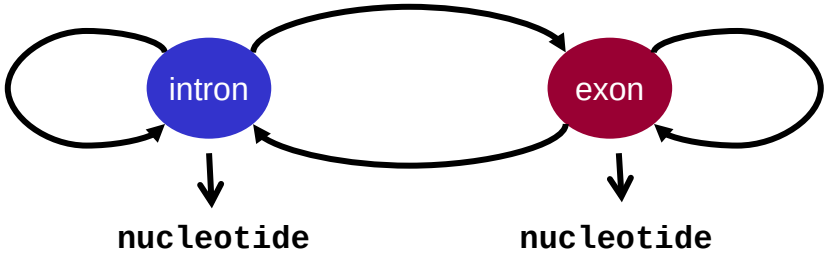
$$\mathbf{B} = \begin{array}{c} \\ \mathtt{I} \\ \mathtt{E} \end{array} \begin{array}{c} \begin{array}{cccc} \mathtt{A} & \mathtt{C} & \mathtt{G} & \mathtt{T} \end{array} \\ \begin{pmatrix} 0.10 & 0.30 & 0.40 & 0.20 \\ 0.50 & 0.20 & 0.20 & 0.10 \end{pmatrix} \end{array}$$

Question

What is the probability of CGA? I.e. P(CGA)?

Likelihood

Consider the following parametrization of the intron-exon example.



$$\mathbf{P} = \begin{array}{c} \\ \mathtt{I} \\ \mathtt{E} \end{array} \begin{array}{c} \begin{array}{cc} \mathtt{I} & \mathtt{E} \end{array} \\ \begin{pmatrix} 0.80 & 0.20 \\ 0.10 & 0.90 \end{pmatrix} \end{array} \quad \text{with} \quad \boldsymbol{\pi} = \begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix} \begin{array}{l} \mathtt{I} \\ \mathtt{E} \end{array}$$

$$\mathbf{B} = \begin{array}{c} \\ \mathtt{I} \\ \mathtt{E} \end{array} \begin{array}{c} \begin{array}{cccc} \mathtt{A} & \mathtt{C} & \mathtt{G} & \mathtt{T} \end{array} \\ \begin{pmatrix} 0.10 & 0.30 & 0.40 & 0.20 \\ 0.50 & 0.20 & 0.20 & 0.10 \end{pmatrix} \end{array}$$

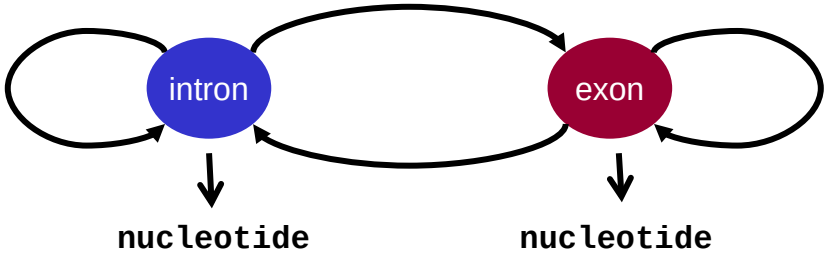
Question

What is the probability of **C** on the 1st position?

I.e. $P(Y_1 = \mathtt{C})$?

Likelihood

Consider the following parametrization of the intron-exon example.



$$\mathbf{P} = \begin{array}{c} \\ \mathtt{I} \\ \mathtt{E} \end{array} \begin{array}{c} \begin{array}{cc} \mathtt{I} & \mathtt{E} \end{array} \\ \begin{pmatrix} 0.80 & 0.20 \\ 0.10 & 0.90 \end{pmatrix} \end{array} \quad \text{with} \quad \boldsymbol{\pi} = \begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix} \begin{array}{l} \mathtt{I} \\ \mathtt{E} \end{array}$$

$$\mathbf{B} = \begin{array}{c} \\ \mathtt{I} \\ \mathtt{E} \end{array} \begin{array}{c} \begin{array}{cccc} \mathtt{A} & \mathtt{C} & \mathtt{G} & \mathtt{T} \end{array} \\ \begin{pmatrix} 0.10 & 0.30 & 0.40 & 0.20 \\ 0.50 & 0.20 & 0.20 & 0.10 \end{pmatrix} \end{array}$$

How likely is the sequence: **CGA**?

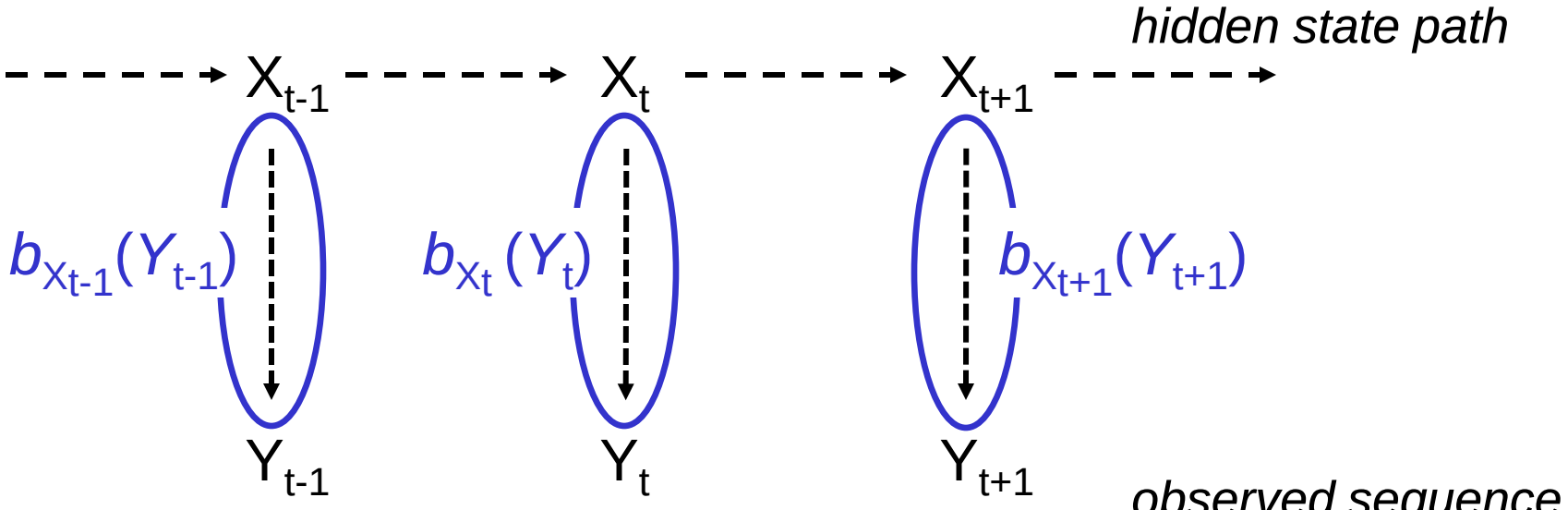
Likelihood

We write down the likelihood of the observed sequence.

Using the conditional independence assumption:

$$P(Y_1, \ldots, Y_T \mid X_1, \ldots, X_T; (\boldsymbol{\pi}, \mathbf{P}, \mathbf{B}))$$

$$= \prod_{t=1}^{T} P(Y_t \mid X_t; (\boldsymbol{\pi}, \mathbf{P}, \mathbf{B})) = \prod_{t=1}^{T} b_{X_t}(Y_t)$$

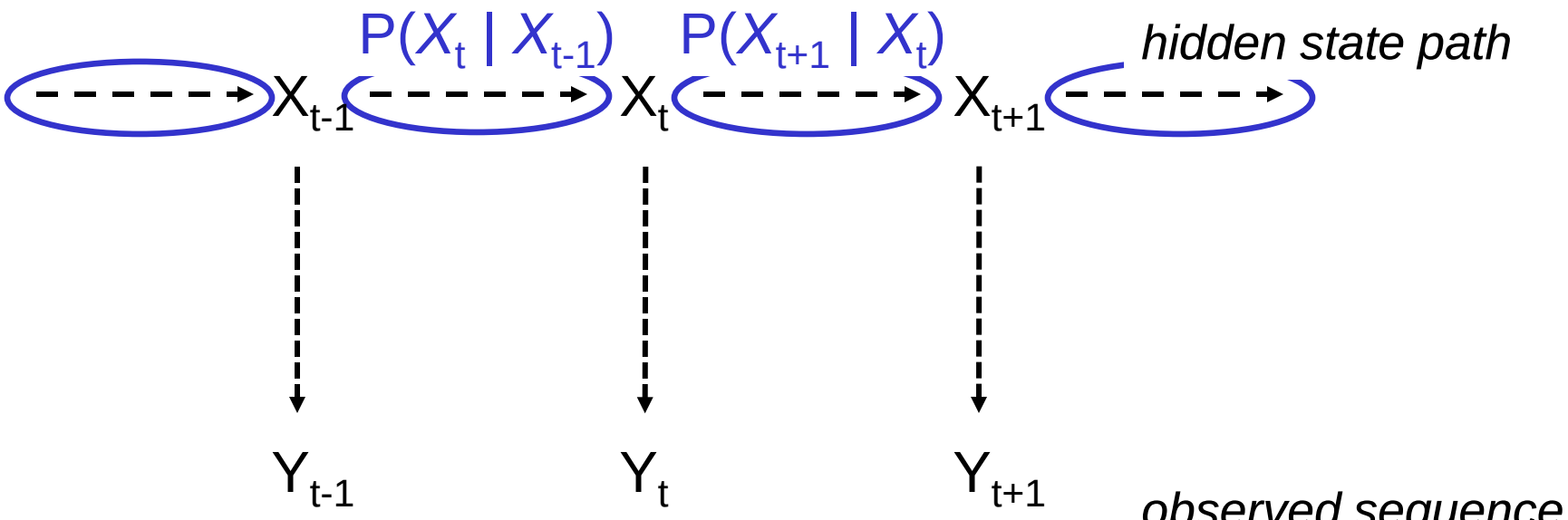


Likelihood

We write down the likelihood of the observed sequence.

Using the Markov property:

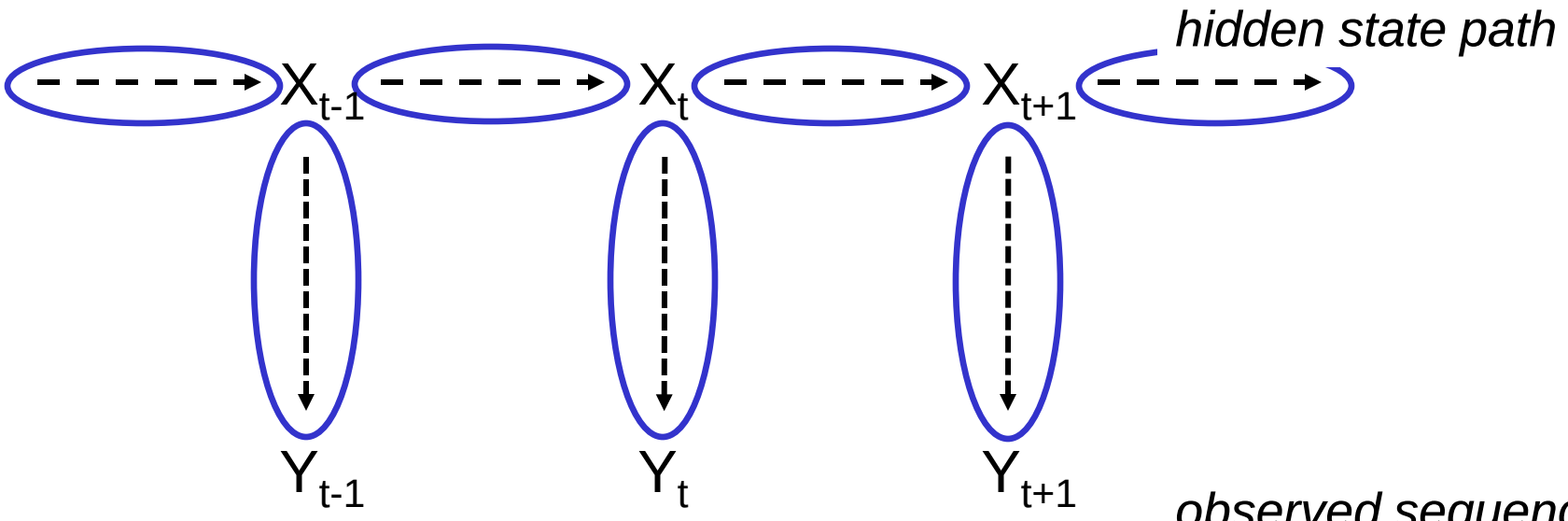
$$P(X_1, \ldots, X_T \,|\, (\boldsymbol{\pi}, \mathbf{P}, \mathbf{B})) \quad = \quad \pi_{X_1} \prod_{t=1}^{T-1} (\mathbf{P})_{X_t; X_{t+1}}$$



Likelihood

Joint distribution $\{X_t\}_{t=1,..,T}$ and $\{Y_t\}_{t=1,..,T}$ is given by:

$$
\begin{aligned}
P(Y_1, \cdots, Y_T, X_1, \ldots, X_T \,|\, (\boldsymbol{\pi}, \mathbf{P}, \mathbf{B})) \\
= \quad P(Y_1, \ldots, Y_T \,|\, X_1, \ldots, X_T; (\boldsymbol{\pi}, \mathbf{P}, \mathbf{B})) \\
\times P(X_1, \ldots, X_T \,|\, (\boldsymbol{\pi}, \mathbf{P}, \mathbf{B}))
\end{aligned}
$$

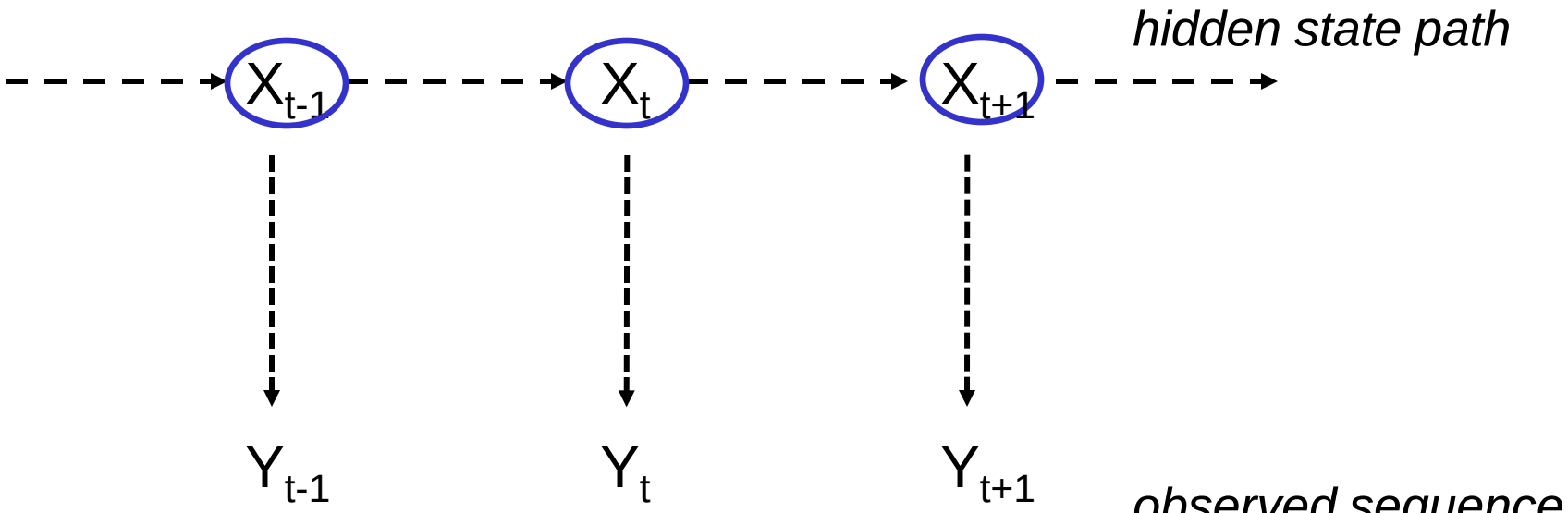


Likelihood

After summing over all possible choices of $\{X_t\}_{t=1,..,T}$, the likelihood of $\{Y_t\}_{t=1,..,T}$ becomes:

$$P(Y_1, \cdots, Y_T \mid (\boldsymbol{\pi}, \mathbf{P}, \mathbf{B}))$$

$$= \sum_{X_1,\ldots,X_T \in S} \pi_{X_1} \prod_{t=1}^{T-1} (\mathbf{P})_{X_t;X_{t+1}} \prod_{t=1}^{T} b_{X_t}(Y_t)$$



Likelihood

Hence, calculate: P((Y_0, Y_1, Y_2) = CGA)

Using the formula on the previous slide:

$$P(Y_0 = \mathtt{C}, Y_1 = \mathtt{G}, Y_2 = \mathtt{A})$$

$$= \sum_{X_0 \in \{\mathtt{I},\mathtt{E}\}} \sum_{X_1 \in \{\mathtt{I},\mathtt{E}\}} \sum_{X_2 \in \{\mathtt{I},\mathtt{E}\}} P(X_0, X_1, X_2)$$

$$\times P(Y_0 = \mathtt{C}, Y_1 = \mathtt{G}, Y_2 = \mathtt{A} \mid X_0, X_1, X_2)$$

$$= \sum_{X_0 \in \{\mathtt{I},\mathtt{E}\}} \sum_{X_1 \in \{\mathtt{I},\mathtt{E}\}} \sum_{X_2 \in \{\mathtt{I},\mathtt{E}\}} P(X_2 \mid X_1) P(X_1 \mid X_0) P(X_0)$$

$$\times P(Y_0 = \mathtt{C} \mid X_0) P(Y_1 = \mathtt{G} \mid X_1) P(Y_2 = \mathtt{A} \mid X_2)$$

Likelihood

Hence, calculate: $P((Y_0, Y_1, Y_2) = \mathtt{CGA})$

Generate all intron-exon sequences of length 3:

III, IIE, IEI, IEE, EEE, EEI, EIE, EII

For each sequence calculate its likelihood, e.g.:

```
P(III) = 0.9 * 0.8 * 0.8 = 0.576
...    = ...             = ...
```

For each sequence calculate the likelihood of **CGA**, e.g.:

```
P(CGA | III) = 0.3 * 0.4 * 0.1 = 0.012
...          = ...             = ...
```

Likelihood

Combine these:

```
P(CGA | III) * P(III) = 0.006912
P(CGA | IIE) * P(IIE) = 0.008640
P(CGA | IEI) * P(IEI) = 0.000096
P(CGA | IEE) * P(IEE) = 0.004860
P(CGA | EEE) * P(EEE) = 0.001620
P(CGA | EEI) * P(EEI) = 0.000036
P(CGA | EIE) * P(EIE) = 0.000080
P(CGA | EII) * P(EII) = 0.000064
```

Sum the above probabilities:

```
P(CGA) = 0.0223
```

HMM vs. Markov chain

HMM vs. Markov chain

All nice these HMMs, but …

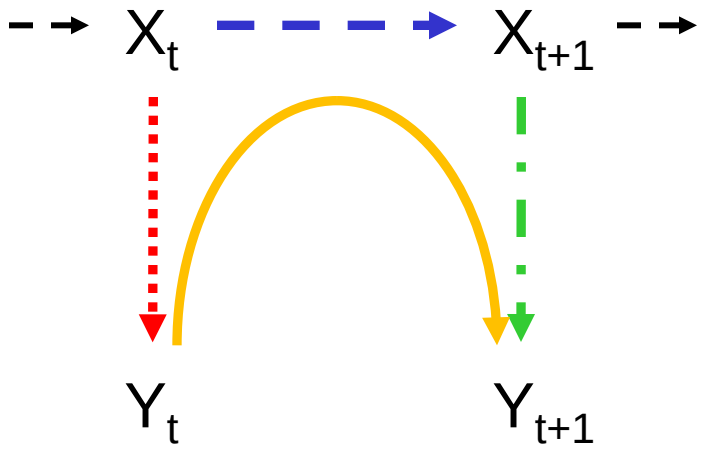
… does it not overly complicate matters?

… can we not make do with a regular Markov model?

A HMM and 1st order Markov chain model the same data:

1st order Markov chain

HMM



Do these models yield identical results?

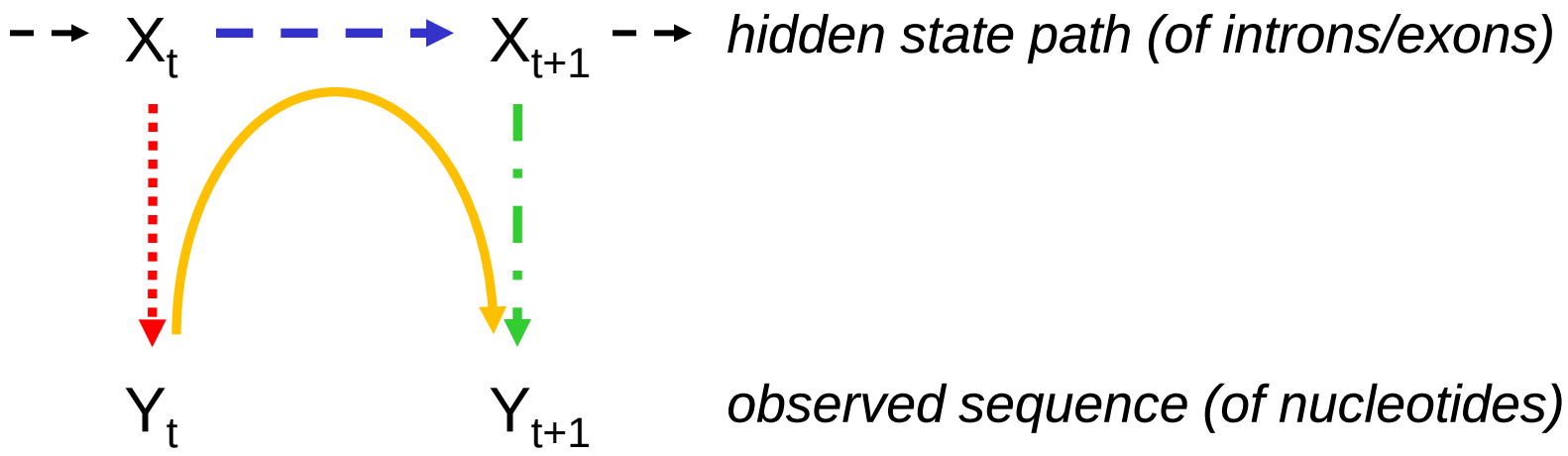
HMM vs. Markov chain

The Markov chain specifies the transition probabilities $P(Y_{t+1} \mid Y_t)$ for the observed sequence.

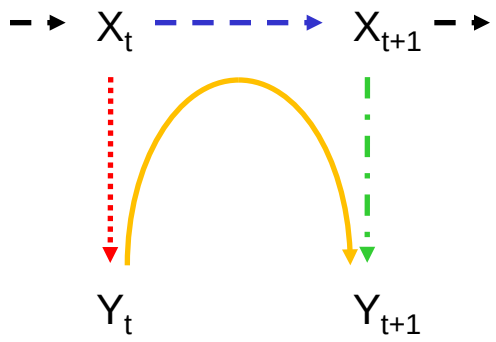
Using the HMM we need to calculate (see next slide):

$$P(Y_{t+1}|Y_t) = \sum_{X_t, X_{t+1}} P(Y_{t+1} \mid X_{t+1})\, P(X_{t+1} \mid X_t)\, P(X_t \mid Y_t)$$

conditional independence



HMM vs. Markov chain



Calculation details previous slide.

$$
\begin{aligned}
P(Y_{t+1} \mid Y_t) & \\
&= \sum_{X_t, X_{t+1}} P(Y_{t+1}, X_t, X_{t+1} \mid Y_t) \\
&= \sum_{X_t, X_{t+1}} P(Y_{t+1} \mid X_t, X_{t+1}, Y_t)\, P(X_t, X_{t+1} \mid Y_t) \\
&= \sum_{X_t, X_{t+1}} P(Y_{t+1} \mid X_{t+1})\, P(X_{t+1} \mid X_t, Y_t)\, P(X_t \mid Y_t) \\
&= \sum_{X_t, X_{t+1}} P(Y_{t+1} \mid X_{t+1})\, P(X_{t+1} \mid X_t)\, P(X_t \mid Y_t)
\end{aligned}
$$

where the definition of conditional probability and the Markov properties of the HMM have been used.

HMM vs. Markov chain

Transition probability:

$$P(Y_{t+1}|Y_t) = \sum_{X_t, X_{t+1}} P(Y_{t+1} \mid X_{t+1})\, P(X_{t+1} \mid X_t)\, P(X_t \mid Y_t)$$

emission matrix of HMM

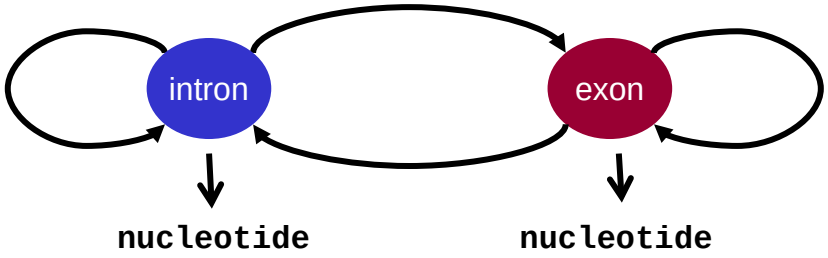
transition matrix of HMM

obtained using Bayes' rule:

$$P(X_t|Y_t) = \frac{P(Y_t \mid X_t)\, P(X_t)}{\sum_{X_t \in \{\mathtt{I},\ \mathtt{E}\}} P(Y_t \mid X_t)\, P(X_t)}$$

HMM vs. Markov chain

Recall the parametrization of the intron-exon example.

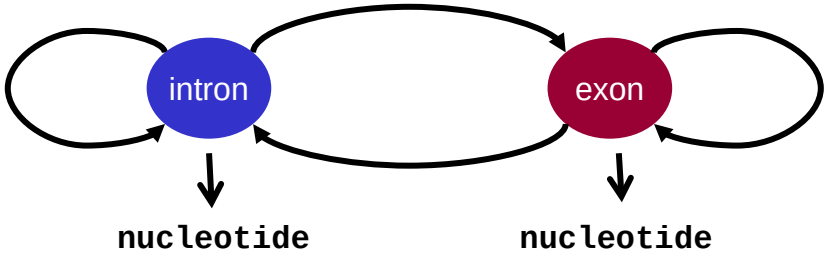


The transition probabilities between the nucleotides using the HMM are (in the form of a transition matrix **P**):

	A	C	G	T
A	0.435	0.216	0.233	0.116
C	0.340	0.240	0.280	0.140
G	0.320	0.245	0.290	0.145
T	0.320	0.245	0.290	0.145

HMM vs. Markov chain

Recall the parametrization of the intron-exon example.



The stationary distribution of this transition matrix

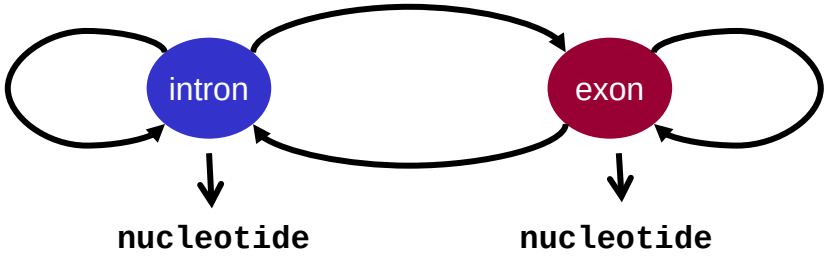
	A	C	G	T
P^n =	0.367	0.233	0.267	0.133

This is the same for the HMM, when we use:

$$\sum_{X^t \in \{\mathtt{I},\mathtt{E}\}} P(X_t)\, b_{X_t}(Y_t) \quad = \quad \psi_{\mathtt{I}}\, b_{\mathtt{I}}(Y_t) + \psi_{\mathtt{E}}\, b_{\mathtt{E}}(Y_t)$$

HMM vs. Markov chain

Recall the parametrization of the intron-exon example.



The initial distribution is sobtained from:

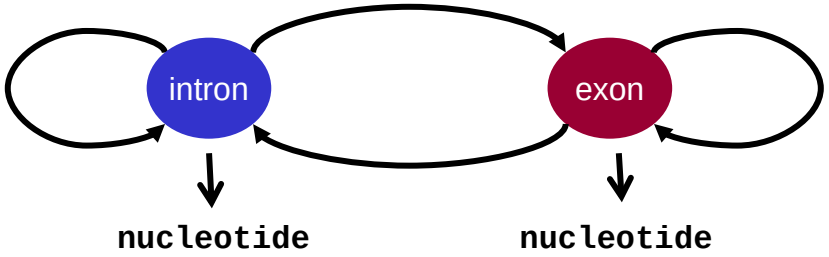
$$P(Y_1) = \sum_{X_1 \in \{\mathtt{I},\ \mathtt{E}\}} P(Y_1 \mid X_1)\, P(X_1)$$

which yields:

	A	C	G	T
π =	0.14	0.29	0.38	0.19

HMM vs. Markov chain

Recall the parametrization of the intron-exon example.



The couple ($\boldsymbol{\pi}$, $\mathbf{P}$) specifies a Markov chain.
However, this Markov chain and the HMM from which it has been derived do not (necessarily) yield the same likelihood:

$$P_{HMM}(CGA) = 0.02232$$

vs.

$$P_{Markov\ chain}(CGA) = 0.025984$$

HMM vs. Markov chain

Parsimony

1st order Markov chain

The transition matrix **P** has 4x4 - 4 free parameters, while for its initial distribution has 4 - 1 parameters. In total: 15 parameters.

HMM

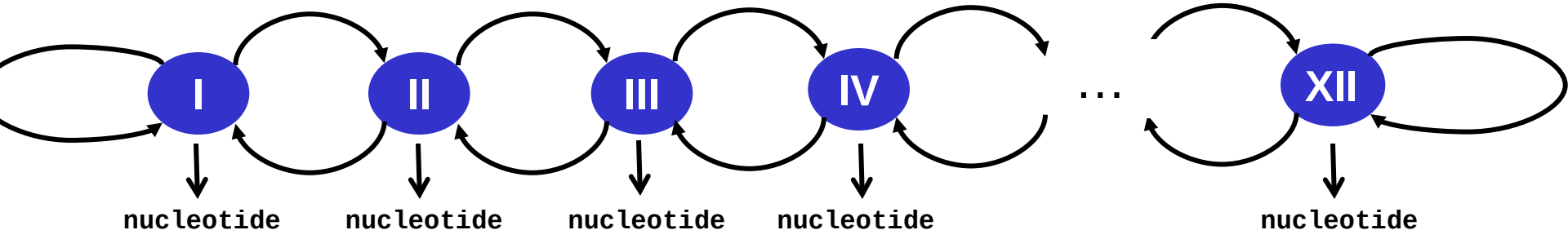
The HMM has 2x2-2 (matrix **P**), 2x4-2 (matrix **B**) and 2-1 (initial distribution) parameters. In total: 9.

The HMM is to be preferred from a parsimony perspective.

HMM vs. Markov chain

Question

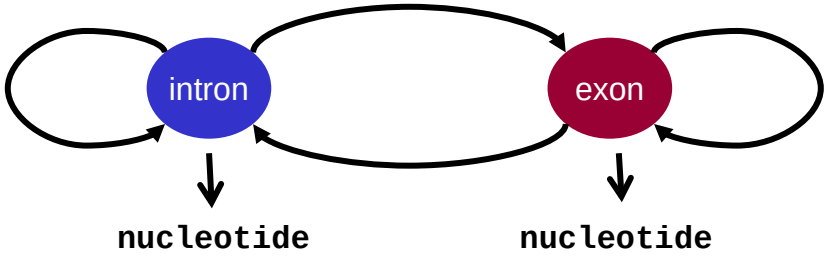
Which model does the parsimony comparison favor when the HMM has a larger latent state space? E.g.:



Could you name other reasons why one model is then preferred?

HMM vs. Markov chain

Recall the parametrization of the intron-exon example.



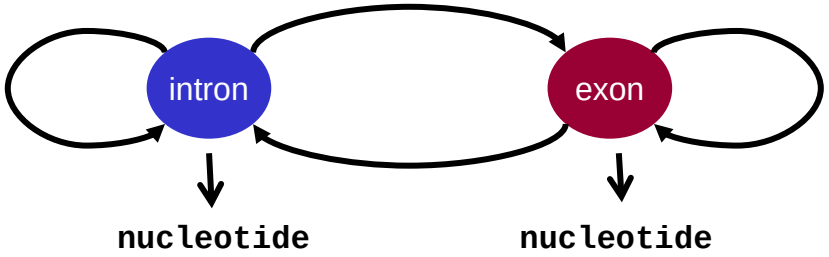
Nonetheless, the HMM *can* be written as a Markov chain with the following state space:

S = {IA, IC, IG, IT, EA, EC, EG, ET}

intron & nucleotide A ... **exon & nucleotide T**

HMM vs. Markov chain

Recall the parametrization of the intron-exon example.



This state space:

$$S = \{IA, IC, IG, IT, EA, EC, EG, ET\}$$

has eight states. Consequently, the corresponding 1st Markov chain has in total: 63 parameters.

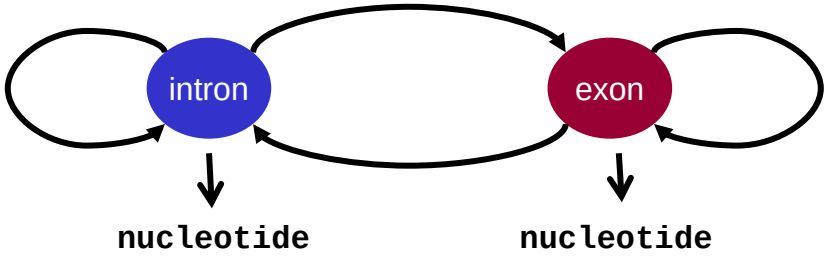
In comparison, the HMM has 9 parameters (in total).

Question

Is this a fair comparison?

HMM vs. Markov chain

Recall the parametrization of the intron-exon example.



The transition matrix of this Markov chain contains, e.g.:

$$P(Y_{t+1}=IA \mid Y_t=EG)$$

To analyze a DNA sequence with Markov chain however requires that *both* the nucleotide and intron/exon sequence have been observed.

Canonical HMM problems

Canonical HMM problems

Three basic problems for HMMs

1) For given parameters $\boldsymbol{\lambda} = (\boldsymbol{\pi}, \mathbf{P}, \mathbf{B})$, how do we calculate $P(\{Y_t\}_{t=1,2,...,T} \mid \boldsymbol{\lambda})$, the probability of an observation, efficiently?

2) For given $\{Y_t\}_{t=1,2,...,T}$ find the sequence $\{X_t\}_{t=1,2,...,T}$ that maximizes $P(\{X_t\}_{t=1,2,...,T} \mid \{Y_t\}_{t=1,2,...,T}, \boldsymbol{\lambda})$. Hence, which underlying state path is most probable?

3) What is the $\boldsymbol{\lambda} = (\boldsymbol{\pi}, \mathbf{P}, \mathbf{B})$ that maximizes the probability of an observed sequence $\{Y_t\}_{t=1,2,...,T}$? How to maximize the likelihood?

Canonical HMM problems

Solutions for the basic problems for HMMs

1) The forward algorithm.

2) The Viterbi algorithm.

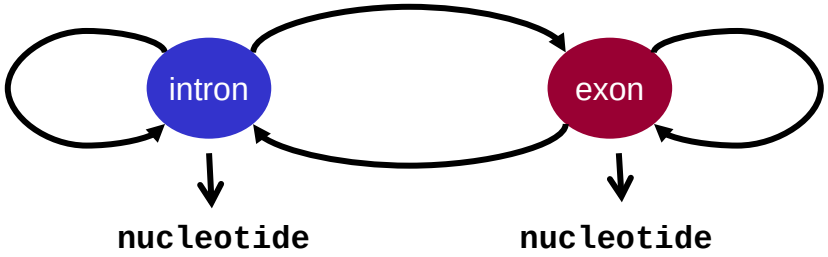
3) The Baum-Welch algorithm *(not discussed)*.

Numbers corresponding to previous slide.

The first two problems could have been solved by direct calculation. This becomes rather cumbersome as T, N and M grow.

The forward algorithm

Consider the following parametrization of the intron-exon example.



$$\mathbf{P} = \begin{array}{c} \\ \mathtt{I} \\ \mathtt{E} \end{array} \begin{array}{c} \begin{array}{cc} \mathtt{I} & \mathtt{E} \end{array} \\ \begin{pmatrix} 0.80 & 0.20 \\ 0.10 & 0.90 \end{pmatrix} \end{array} \quad \text{with} \quad \boldsymbol{\pi} = \begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix} \begin{array}{l} \mathtt{I} \\ \mathtt{E} \end{array}$$

$$\mathbf{B} = \begin{array}{c} \\ \mathtt{I} \\ \mathtt{E} \end{array} \begin{array}{c} \begin{array}{cccc} \mathtt{A} & \mathtt{C} & \mathtt{G} & \mathtt{T} \end{array} \\ \begin{pmatrix} 0.10 & 0.30 & 0.40 & 0.20 \\ 0.50 & 0.20 & 0.20 & 0.10 \end{pmatrix} \end{array}$$

How likely is the sequence: **CGA**?

The forward algorithm

Hence, calculate: $P((Y_0, Y_1, Y_2) = \mathbf{CGA})$

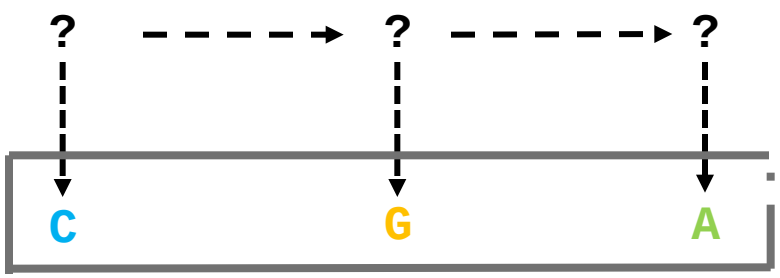
- Generate all intron-exon sequences of length 3.
  III, IIE, IEI, IEE, EEE, EEI, EIE, EII
- For each sequence calculate its likelihood, e.g.:
  P(III) = 0.9 * 0.8 * 0.8 = 0.576
- For each sequence calculate the likelihood of **CGA,** e.g.:
  P(CGA | III) = 0.3 * 0.4 * 0.1 = 0.012
- Combine these:
  P(CGA | III) * P(III) = 0.006912
- Sum the above probabilities:
  P(CGA) = 0.0223

Impractical for longer sequences: *forward algorithm*!

The forward algorithm

Idea behind forward algorithm

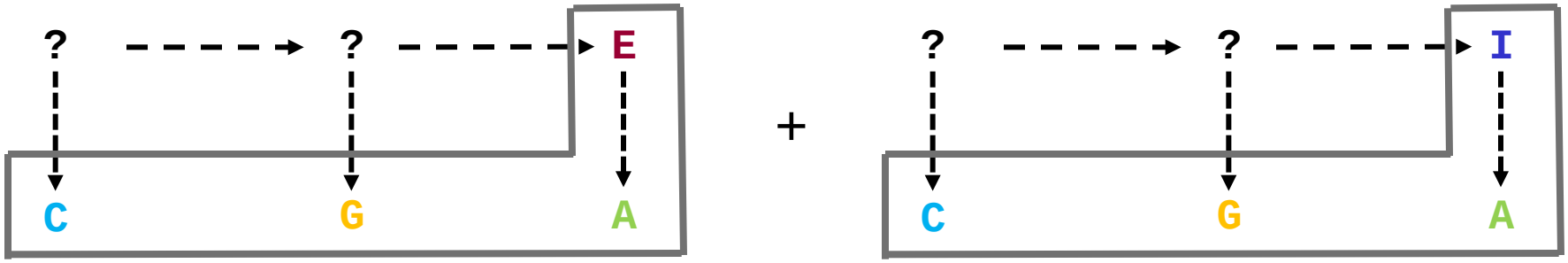
Likelihood of **CGA**:



hidden state path

observed sequence

equals (recall: total probability law)



These two quantities are “easy” to calculate.

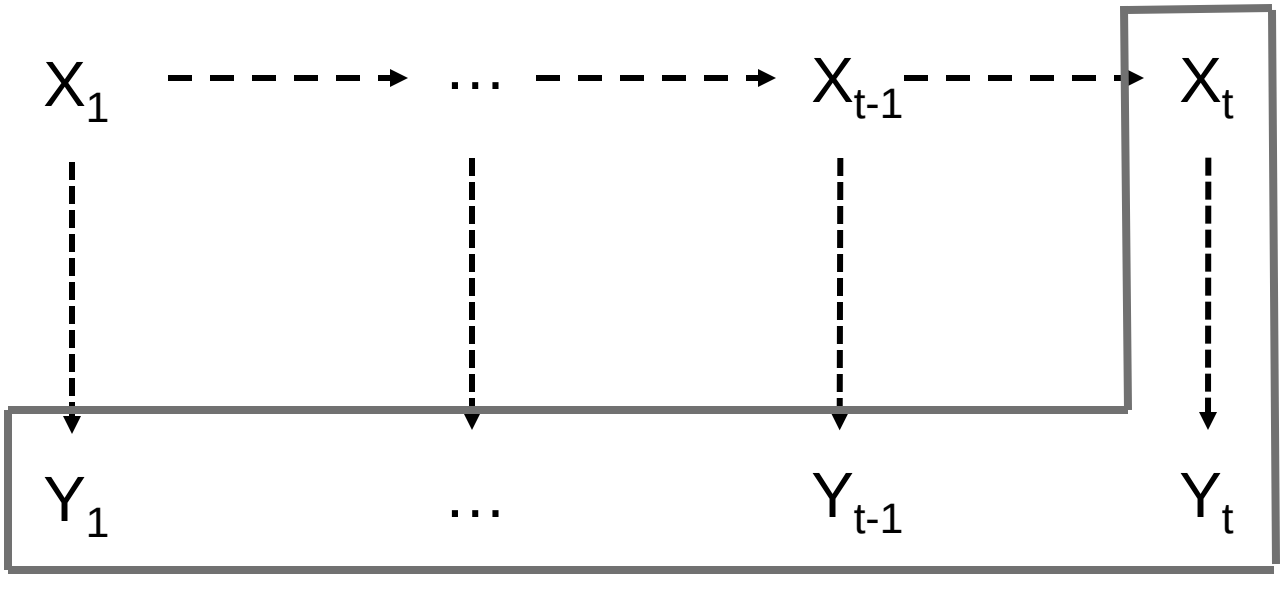
The forward algorithm

book Axelson-Fisk (2010): Section 2.1.4

Here we illustrate the calculation of $P(Y_1, ..., Y_T \mid \boldsymbol{\lambda})$ using the forward algorithm.

First define the *forward variable*:

$$\alpha_t(i) \quad = \quad P(Y_1, \ldots, Y_t, X_t = S_i \mid \boldsymbol{\lambda})$$



hidden state path

observed sequence

The forward algorithm

This forward variable is of interest for:

$$\sum_{i=1}^{N} \alpha_T(i) = \sum_{i=1}^{N} P(Y_1, \ldots, Y_T, X_T = S_i \,|\, \boldsymbol{\lambda})$$
$$= P(Y_1, \ldots, Y_T \,|\, \boldsymbol{\lambda})$$

This uses total probability law, summing over all possible states of the unobserved hidden variable at a particular time.

In order to compute $P(Y_1, \ldots, Y_T \,|\, \boldsymbol{\lambda})$ efficiently, we may derive (see SM) a recursive relationship for the forward variable:

$$\alpha_t(i) = b_i(Y_t) \sum_{j=1}^{N} \alpha_{t-1}(j) \, (\mathbf{P})_{ji}$$

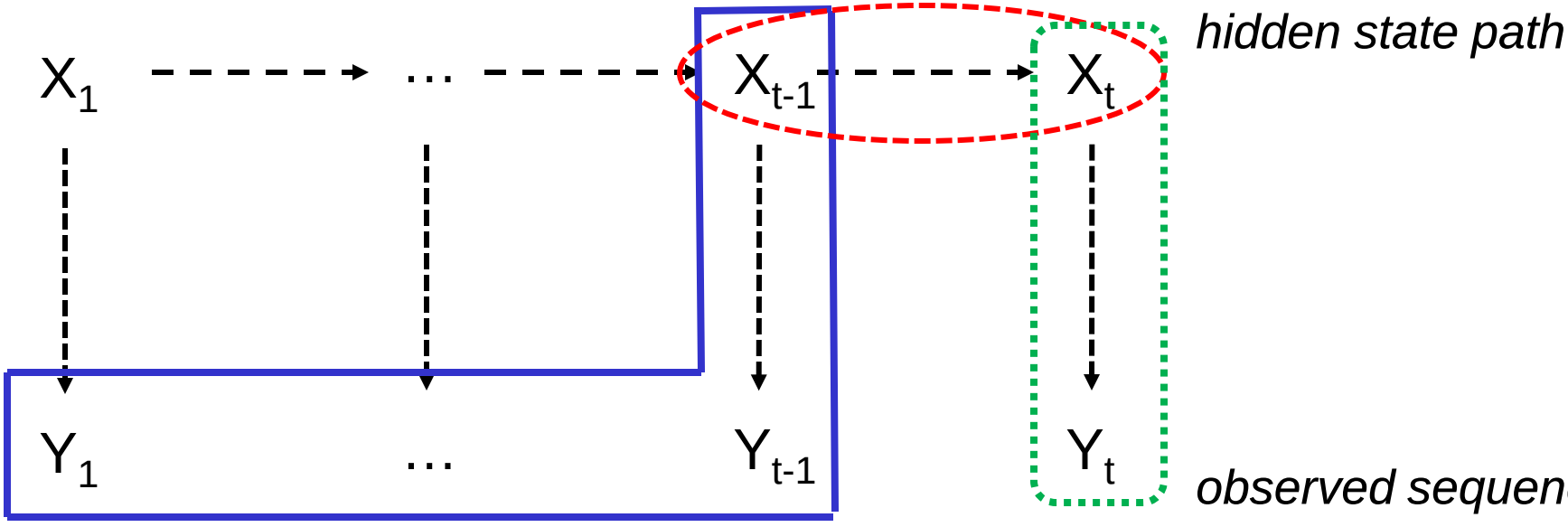
The forward algorithm

book Axelson-Fisk (2010): Section 2.1.4

The recursive relationship for the forward variable (graphically)

$$\alpha_t(i) = b_i(Y_t) \sum_{j=1}^{N} \alpha_{t-1}(j) (\mathbf{P})_{ji}$$

The sum over j washes out X_{t-1}.



The forward algorithm

Apply the recursive relationship:

$$\alpha_t(i) = b_i(Y_t) \sum_{j=1}^{N} \alpha_{t-1}(j) (\mathbf{P})_{ji}$$

to the example:

$P((Y_1, Y_2, Y_3)=(\mathbf{C}, \mathbf{G}, \mathbf{A}), X_3=\mathbf{E}) =$

$\alpha_3(\mathtt{E})$

$P(Y_3=\mathbf{A} \mid X_3=\mathbf{E}) * [P((Y_1, Y_2)=(\mathbf{C}, \mathbf{G}), X_2=\mathbf{E}) * P(X_3=\mathbf{E} \mid X_2=\mathbf{E})$

$b_{\mathtt{E}}(\mathtt{A})$ $\alpha_2(\mathtt{E})$ $\mathbf{P}_{(\mathtt{E},\mathtt{E})}$

$+ P((Y_1, Y_2)=(\mathbf{C}, \mathbf{G}), X_2=\mathbf{I}) * P(X_3=\mathbf{E} \mid X_2=\mathbf{I})]$

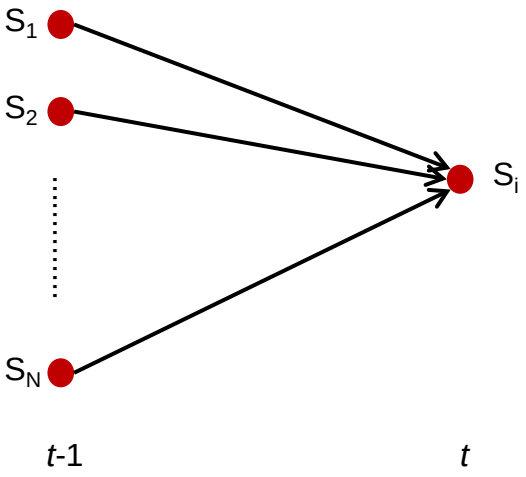
$\alpha_2(\mathtt{I})$ $\mathbf{P}_{(\mathtt{I},\mathtt{E})}$

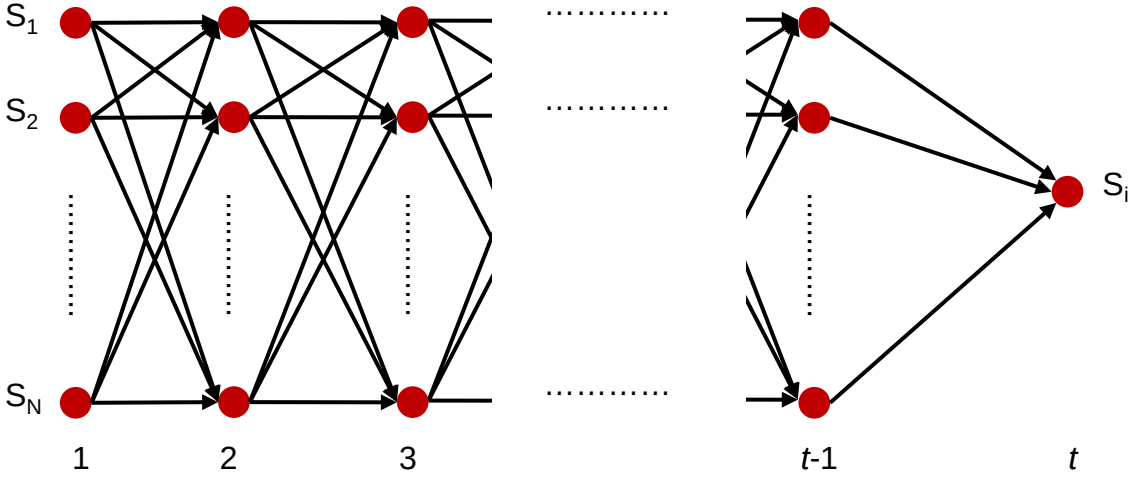
The forward algorithm

The recursive relationship expresses the forward variable at time *t* in terms of those at time *t*-1.

Walking forward in time, to calculate the probability of interest we only store the last probabilities.

Dynamic programming!





The forward algorithm

The forward algorithm comprises of:

1) *Initialization*:

$$\alpha_1(i) \quad = \quad \pi_i \, b_i(Y_1)$$

2) *Induction*:

$$\alpha_t(i) \quad = \quad b_i(Y_t) \sum_{j=1}^{N} \alpha_{t-1}(j) \, (\mathbf{P})_{ji}$$

$$\text{for} \quad 2 \leq t \leq T, 1 \leq i \leq N$$

3) *Termination*:

$$P(Y_1, \ldots, Y_T \,|\, \boldsymbol{\lambda}) \quad = \quad \sum_{i=1}^{N} \alpha_T(i)$$

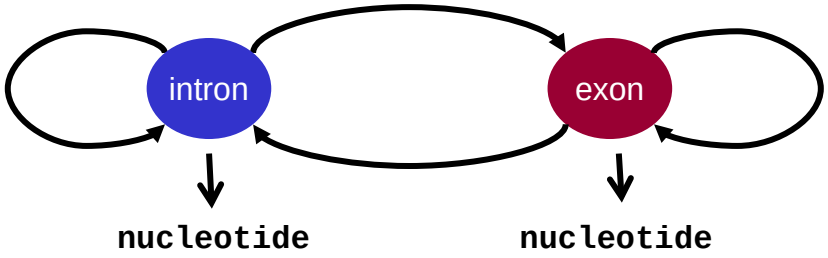
The forward algorithm

The forward algorithm comprises of:

1) *Initialization*:

$$\alpha_1(i) \quad = \quad \pi_i \, b_i(Y_1)$$

Reconsider the intron-exon example.



How likely is the sequence: **CGA**?

Initialization:

$\alpha_1(I) = 0.9 * 0.3 = 0.27$

$\alpha_1(E) = 0.1 * 0.2 = 0.02$

The forward algorithm

The forward algorithm comprises of:

2) Induction:

$$\alpha_t(i) = b_i(Y_t) \sum_{j=1}^{N} \alpha_{t-1}(j) \, (\mathbf{P})_{ji} \quad \text{for} \quad 2 \leq t \leq T, 1 \leq i \leq N$$

Induction:

$\alpha_2(I) = 0.4 * (0.27 * 0.8 + 0.02 * 0.1)$
$= 0.0872$
$\alpha_2(E) = 0.2 * (0.27 * 0.2 + 0.02 * 0.9)$
$= 0.0144$

$\alpha_3(I) = 0.1 * (0.0872 * 0.8 + 0.0144 * 0.1)$
$= 0.00712$
$\alpha_3(E) = 0.5 * (0.0872 * 0.2 + 0.0144 * 0.9)$
$= 0.0152$

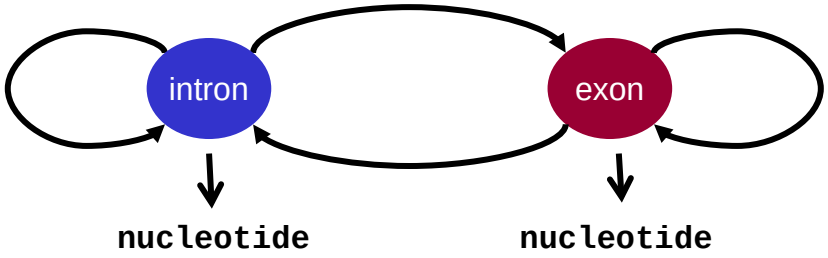
The forward algorithm

The forward algorithm comprises of:

3) Termination:

$$P(Y_1, \ldots, Y_T \mid \boldsymbol{\lambda}) = \sum_{i=1}^{N} \alpha_T(i)$$

Reconsider the intron-exon example.



Termination:

α_3(I) + α_3(E) = 0.00712 + 0.0152
= 0.0223

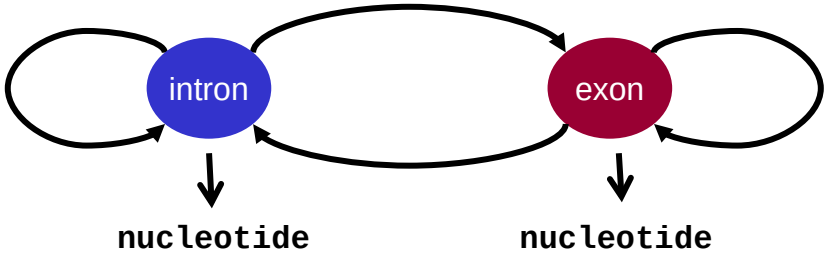
As before:

P(CGA) = 0.0223

The Viterbi algorithm

The Viterbi algorithm

Consider the following parametrization of the intron-exon sample.



$$\mathbf{P} = \begin{array}{c} \\ \mathtt{I} \\ \mathtt{E} \end{array} \begin{array}{c} \begin{array}{cc} \mathtt{I} & \mathtt{E} \end{array} \\ \begin{pmatrix} 0.80 & 0.20 \\ 0.10 & 0.90 \end{pmatrix} \end{array} \quad \text{with} \quad \boldsymbol{\pi} = \begin{array}{c} \\ \mathtt{I} \\ \mathtt{E} \end{array} \begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix}$$

$$\mathbf{B} = \begin{array}{c} \\ \mathtt{I} \\ \mathtt{E} \end{array} \begin{array}{c} \begin{array}{cccc} \mathtt{A} & \mathtt{C} & \mathtt{G} & \mathtt{T} \end{array} \\ \begin{pmatrix} 0.10 & 0.30 & 0.40 & 0.20 \\ 0.50 & 0.20 & 0.20 & 0.10 \end{pmatrix} \end{array}$$

What is the most likely intron-exon sequence to have generated the nucleotide sequence: **CGA**?

The Viterbi algorithm

What is the most likely intron-exon sequence to have generated the nucleotide sequence: **CGA**?

Question

How would you proceed to answer the above?

→ Would you base your decision on (e.g.):
P(IIE|CGA), P(CGA|IIE), P(IIE), or **P(CGA)?**

→ How do you distinguish between (say) **IIE** and **EIE**?

The Viterbi algorithm

Hence, we need to calculate:

$$\max_{X_0,X_1,X_2} P((X_0,X_1,X_2) \mid (Y_0,Y_1,Y_2))$$

$$= \max_{X_0,X_1,X_2} P((Y_0,Y_1,Y_2) \mid (X_0,X_1,X_2)) \times P((X_0,X_1,X_2))/P((Y_0,Y_1,Y_2))$$

Bayes' rule

Intron/exon example, for any choice of (X_0, X_1, X_2) evaluate:

P((X_0, X_1, X_2) | (Y_0, Y_1, Y_2) = **CGA**)

The Viterbi algorithm

Let us calculate, e.g:

$P((X_0, X_1, X_2) = \mathbf{III} \mid (Y_0, Y_1, Y_2) = \mathbf{CGA})$

The likelihood of the latent intron-exon sequence is:

P(III) = 0.9 * 0.8 * 0.8 = 0.576

Given the intron-exon sequence calculate the likelihood of the observed nucleotide sequence **CGA**:

P(CGA | III) = 0.3 * 0.4 * 0.1 = 0.012

Also, calculate the likelihood of the observed nucleotide sequence **CGA** using the forward algorithm:

P(CGA) = 0.0223

The Viterbi algorithm

Calculate all conditional probabilities:

```
P(III | CGA) = P(CGA | III) * P(III) / P(CGA)
             = 0.006912 / 0.0223 = 0.310
P(IIE | CGA) = ...                 = 0.387
P(IEI | CGA) = ...                 = 0.004
P(IEE | CGA) = ...                 = 0.218
P(EEE | CGA) = ...                 = 0.073
P(EEI | CGA) = ...                 = 0.002
P(EIE | CGA) = ...                 = 0.004
P(EII | CGA) = ...                 = 0.003
```

Select the one with the highest probability:

IIE

Q: Necessary to calculate the likelihood?

The Viterbi algorithm

The preceeding calculation is impractical for long sequences.

The Viterbi algorithm efficiently calculates:

$$P(X_1, ..., X_T \mid Y_1, ..., Y_T, \boldsymbol{\lambda}).$$

First, define the variable:

$$\delta_t(i) = \max_{X_1,\ldots,X_{t-1}} P(X_1, \ldots, X_{t-1}, X_t = S_i, Y_1, \ldots, Y_t \mid \boldsymbol{\lambda})$$

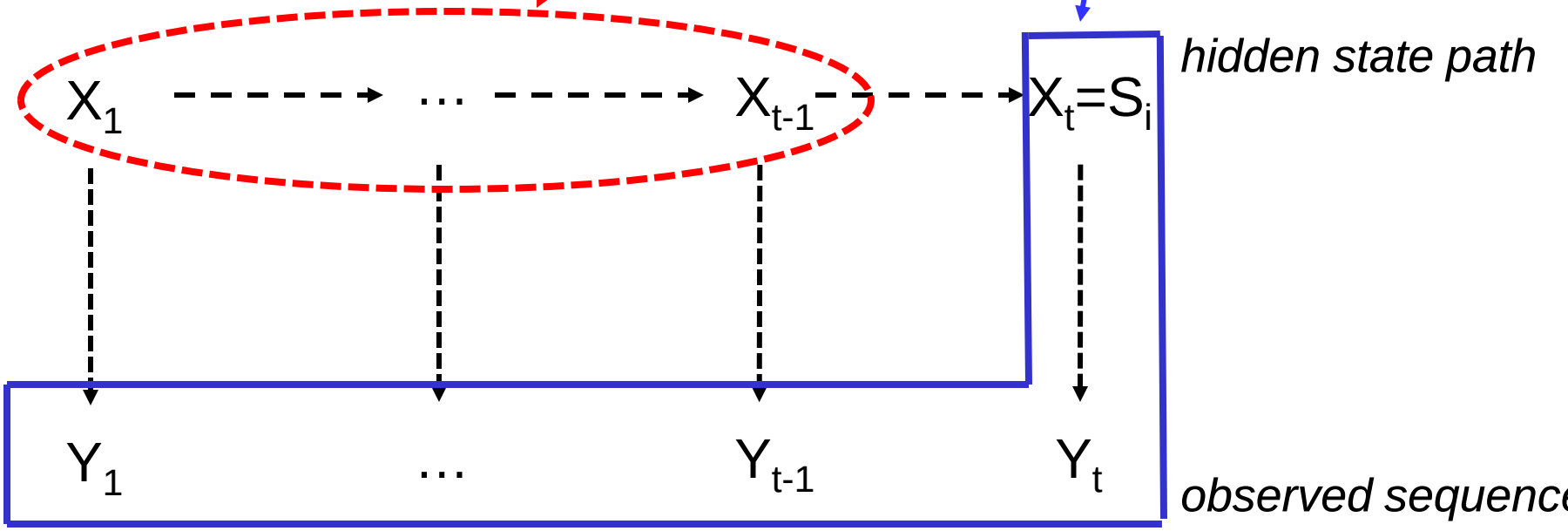
- The maximum is over all paths of length *t* ending in state S_i.
- The likelihood of this path together with the *t* associated observations is maximized.

Note:

$$P(X_1, ..., X_T \mid Y_1, ..., Y_T, \boldsymbol{\lambda}) = \delta_t(i) \,/\, \underbrace{P(Y_1, ..., Y_T, \boldsymbol{\lambda})}_{\text{likelihood}}$$

The Viterbi algorithm

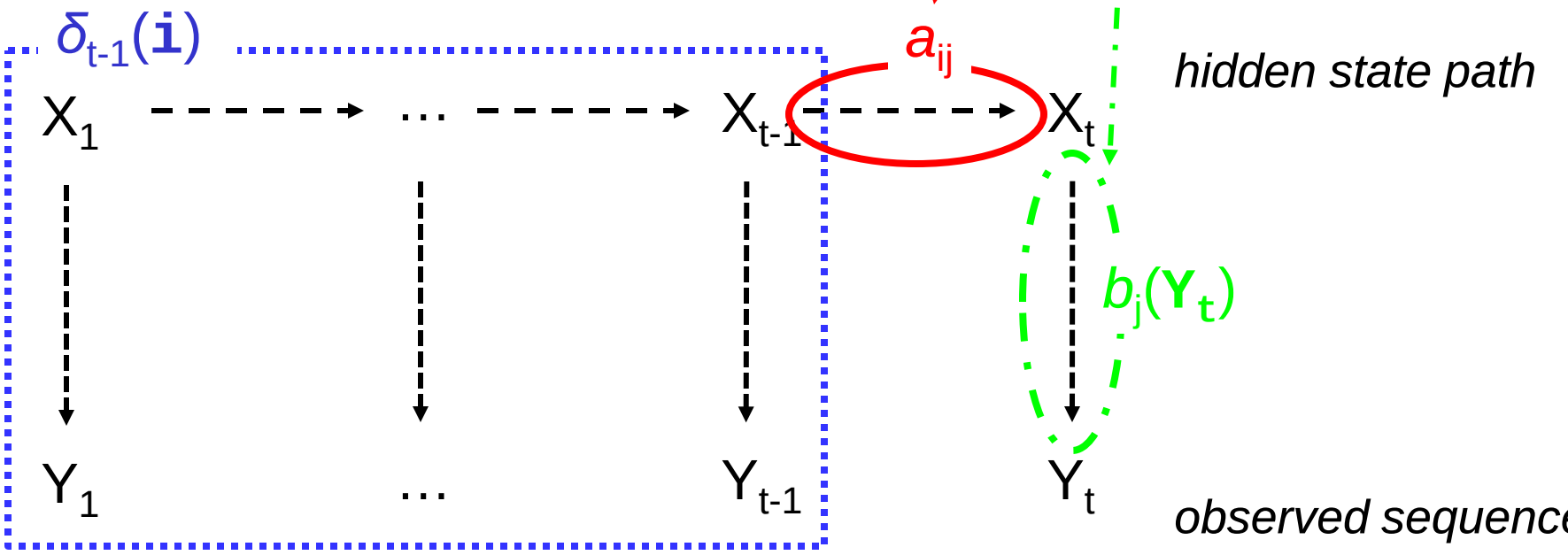
$$\delta_t(i) = \max_{X_1,\ldots,X_{t-1}} P(X_1,\ldots,X_{t-1}, X_t = S_i, Y_1,\ldots,Y_t \mid \boldsymbol{\lambda})$$



The Viterbi algorithm

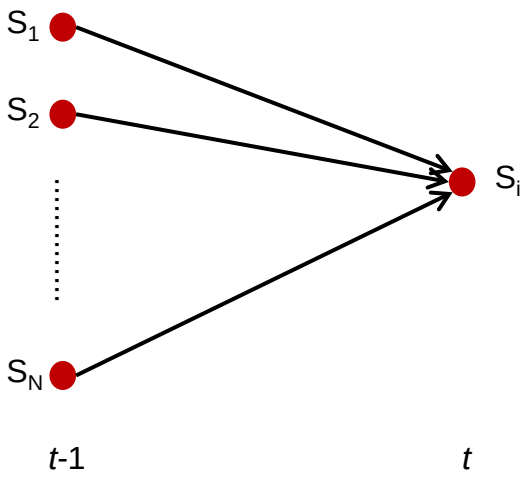
The recursive relationship:

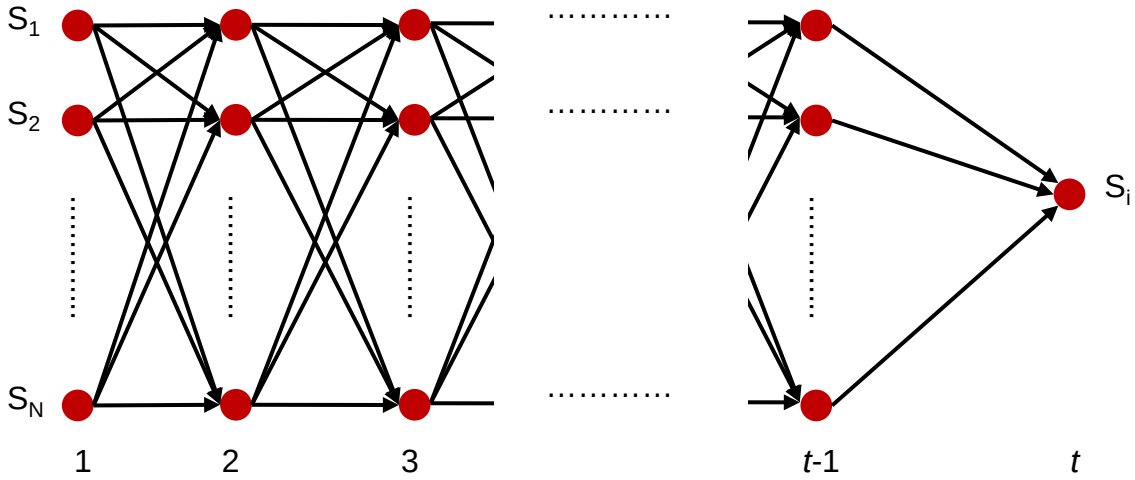
$$\delta_t(j) = [\max_i \delta_{t-1}(i) a_{ij}] \, b_j(Y_t)$$



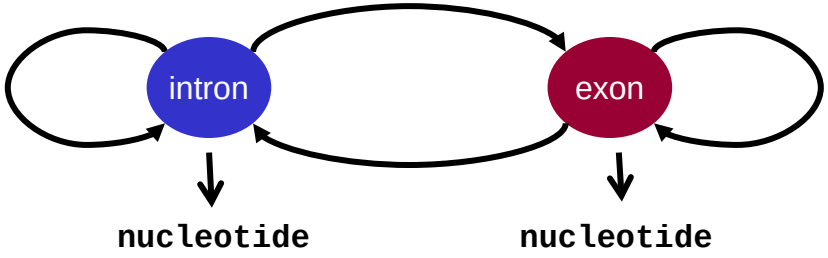
The Viterbi algorithm

As with the forward variable, the recursive relationship expresses the ‘Viterbi’ variable at time *t* in terms of those at time *t*-1. Hence, we do not need to keep track of probabilities at all time points.





The Viterbi algorithm



The Viterbi algorithm comprises of:

1) Initialization:

$$\delta_1(j) = \pi_j \, b_j(Y_1) \qquad 1 \leq j \leq N$$
$$\psi_1(j) = 0$$

Initiation of Viterbi:

$\log[\delta_1(I)] = \log[P(X_1=I)] + \log[P(Y_1=C \mid X_1=I)]$
$= \log(0.9) + \log(0.3)$
$= -1.309333$

$\log[\delta_1(E)] = -3.912023$

The log-scale is used for efficiency and accuracy.

The Viterbi algorithm

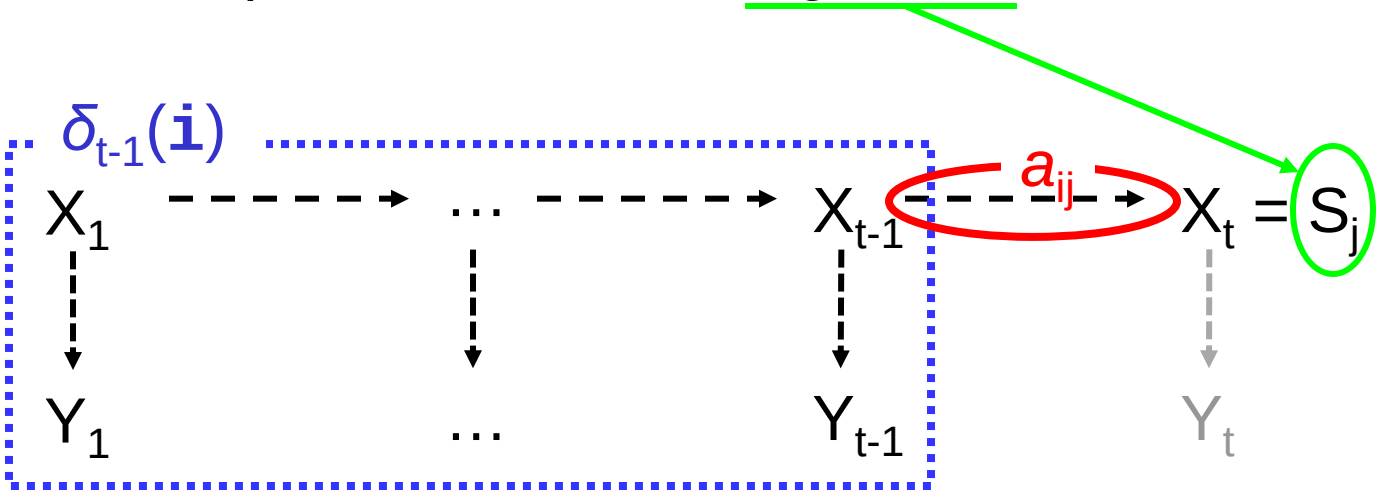
The Viterbi algorithm comprises of:

2) Induction:

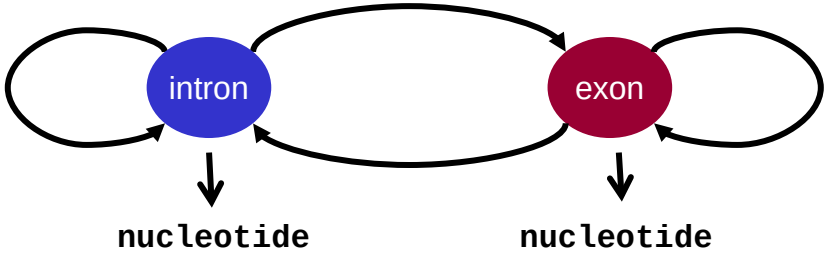
$$\delta_t(j) = \max_{1 \le i \le N} \delta_{t-1}(i)\, a_{ij}\, b_j(Y_t) \qquad 1 \le j \le N$$

$$\psi_t(j) = \arg\max_{1 \le i \le N} \delta_{t-1}(i)\, a_{ij} \qquad 2 \le t \le T$$

$\psi_t(j)$ keeps track of which argument maximized $\delta_t(j)$



The Viterbi algorithm



Induction (first step only):

$$\log[\delta_2(I)]$$
$$= \log[P(Y_2=G \mid X_2=I)] + \max\{[\log[\delta_1(X_1)] + \log[P(X_2=I \mid X_1)]\}$$
$$= \log(0.4) + \max\{-1.309333 + \log(0.8), -3.912023 + \log(0.1)\}$$
$$= -0.9162907 + \max\{-1.532477, -6.214608\}$$
$$= -2.448768$$

Similarly:

$$\log[\delta_2(E)] = -4.528209$$

$$\log[\delta_3(I)] = -4.974497$$

$$\log[\delta_3(E)] = -4.751353$$

The Viterbi algorithm

Parallel we get:

$$\Psi_1(I) = 0,\ \Psi_1(E) = 0$$

$$\Psi_2(I) = \arg\max_{X_1} \{\delta_1(X_1=I) * P(X_2=I \mid X_1=I,\ \delta_1(X_1=E) * P(X_2=I \mid X_1=E)\}$$

$$= \arg\max_{X_1} \{\exp(-1.309333) * 0.80,\ \exp(-3.912023) * 0.10\ \}$$

$$= I$$

$$\Psi_2(E) = \arg\max_{X_1} \{\delta_1(X_1=I) * P(X_2=E \mid X_1=I),\ \delta_1(X_1=E) * P(X_2=E \mid X_1=E)\}$$

$$= \arg\max_{X_1} \{\exp(-1.309333) * 0.20,\ \exp(-3.912023) * 0.90\ \}$$

$$= I$$

The Viterbi algorithm

Parallel we get:

$$\Psi_3(I) = \arg\max_{X_2} \{\delta_2(X_1=I) * P(X_3=I \mid X_2=I), \delta_2(X_1=E) * P(X_3=I \mid X_2=E)\}$$
$$= \arg\max_{X_2} \{\exp(-2.448768) * 0.80, \exp(-4.528209) * 0.10\}$$
$$= I$$

$$\Psi_3(E) = \arg\max_{X_2} \{\delta_2(X_2=I) * P(X_3=E \mid X_2=I), \delta_2(X_2=E) * P(X_3=E \mid X_2=E)\}$$
$$= I$$

Similarly:

$$\Psi_4(I) = I$$

$$\Psi_4(E) = E$$

The Viterbi algorithm

Results from induction step:

	Y_1 = C	Y_2 = G	Y_3 = A
	δ_1	δ_2	δ_3
Intron	-1.309333	-2.448768	-4.974497
Exon	-3.912023	-4.528209	-4.751353

	ψ_2	ψ_3	ψ_4
Intron	I	I	I
Exon	I	I	E

The Viterbi algorithm

The Viterbi algorithm finalized with:

3) Termination:

$$p^* = \max_{1 \leq i \leq N} \delta_T(i)$$

$$q_T^* = \arg \max_{1 \leq i \leq N} \delta_T(i)$$

This yields:

	Y_1 = C	Y_2 = G	Y_3 = A
Intron	-1.309333	-2.448768	-4.974497
Exon	-3.912023	-4.528209	-4.751353

p*

Thus: q_3^* = E

The Viterbi algorithm

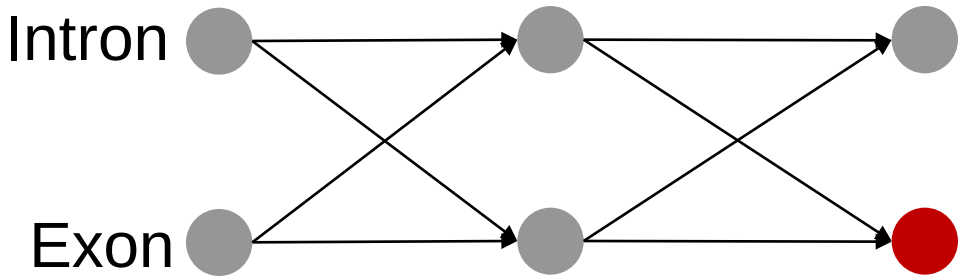
The Viterbi algorithm finalized with:

4) Backtracking:

$$q^*_{t-1} = \psi_t(q^*_t) \qquad t = T, T-1, \ldots, 2$$

Backtracking:

an underlying sequence with an **exon** at *t*=3 is most likely.

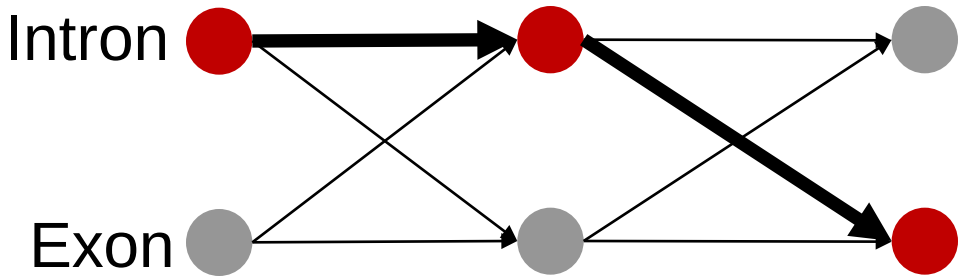


The Viterbi algorithm

Continue backtracking:

	Y_1 = C	Y_2 = G	Y_3 = A
Intron	-1.309333	-2.448768	-4.974497
Exon	-3.912023	-4.528209	-4.751353

Thus: ψ_3(E) = I and ψ_2(I) = I

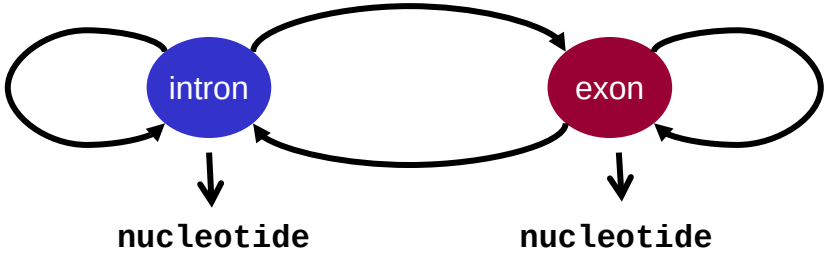


The most likely underlying sequence is: **IIE**

The forward algorithm

Illustration

Sample from the following parametrization of the intron-exon example.



$$\mathbf{P} = \begin{array}{c} \\ \mathtt{I} \\ \mathtt{E} \end{array} \begin{array}{c} \begin{array}{cc} \mathtt{I} & \mathtt{E} \end{array} \\ \begin{pmatrix} 0.80 & 0.20 \\ 0.10 & 0.90 \end{pmatrix} \end{array} \quad \text{with} \quad \boldsymbol{\pi} = \begin{array}{c} \\ \mathtt{I} \\ \mathtt{E} \end{array} \begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix}$$

$$\mathbf{B} = \begin{array}{c} \\ \mathtt{I} \\ \mathtt{E} \end{array} \begin{array}{c} \begin{array}{cccc} \mathtt{A} & \mathtt{C} & \mathtt{G} & \mathtt{T} \end{array} \\ \begin{pmatrix} 0.10 & 0.30 & 0.40 & 0.20 \\ 0.50 & 0.20 & 0.20 & 0.10 \end{pmatrix} \end{array}$$

Data:

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

The Viterbi algorithm

Illustration

Most likely underlying state sequence? → Viterbi!

True states (1st 100 positions)

IIIIIIEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEIIIIIIIIIIIIIIIIEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE

Inferred states (1st 100 positions)

IIIIIIIEEEEEEEEEEIIIIIEEEEEEEEEEEEEEEEEEIIIIIIIIIIIIEEEEEEEEEIIIEEEEIIEEEEEEEEEEEEEEIIIEEEEEEEE

True vs. inferred (10000 positions):

	exon	intron (true)
exon	5655	1633
intron	776	1936

The Viterbi algorithm

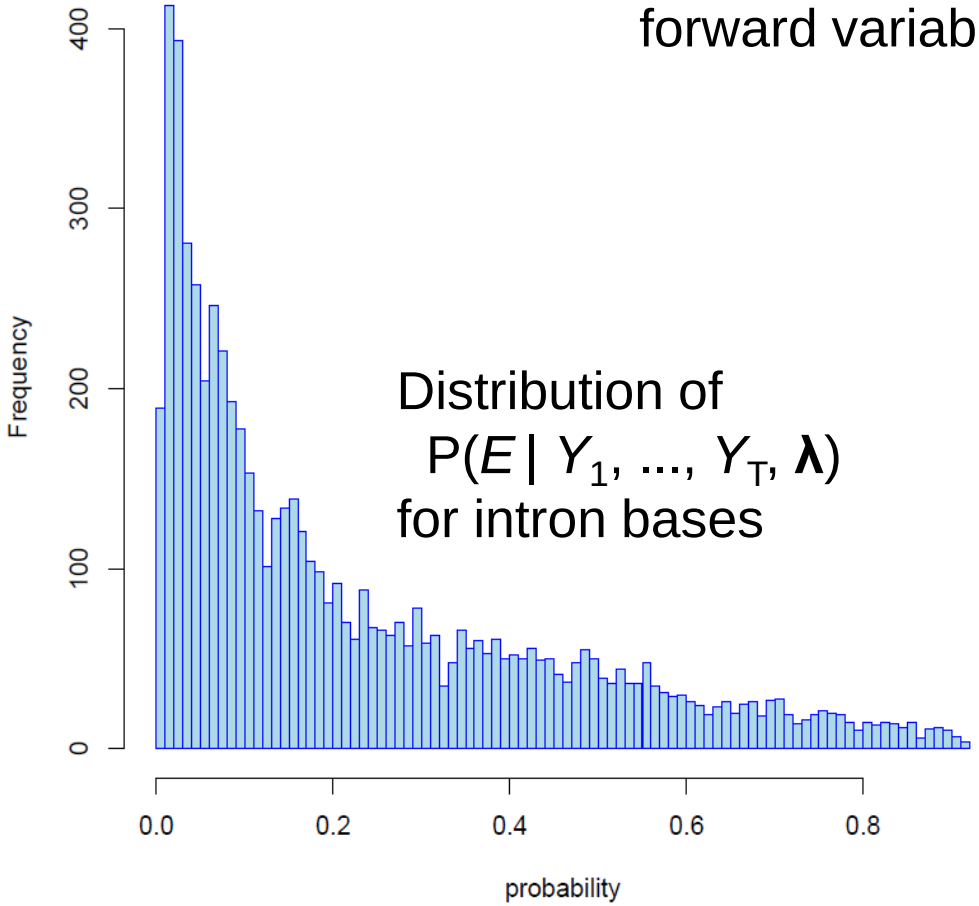
Illustration

Related is the posterior probability of states:

$$P(X_t \mid Y_1, \ldots, Y_T, \boldsymbol{\lambda}) = P(X_t, Y_1, \ldots, Y_T, \boldsymbol{\lambda}) \,/\, P(Y_1, \ldots, Y_T, \boldsymbol{\lambda}).$$

forward variable

likelihood



The Viterbi algorithm

```
# load library
library(HMM)

# specify HMM
hmm <- initHMM(c("I","E"),c("A","C","G", "T"),
       transProbs=matrix(c(0.8, 0.2, 0.1, 0.9), 2, 2, byrow=TRUE),
       emissionProbs=matrix(c(0.1, 0.3, 0.4, 0.2, 0.5, 0.2, 0.2, 0.1),
       2, 4, byrow=TRUE))

# simulate from the HMM
seqLength <- 10000
DNAseq <- simHMM(hmm, seqLength)

# apply viterbi and compare to true underlying sequence
table(viterbi(hmm, DNAseq$observation), DNAseq$states)

# obtain posterior probabilities of latent states (given the sequence)
posterior(hmm, DNAseq$observation)

# histogram of post. probabilities for exons.
hist(posterior(hmm, obsSequence$observation)[1,obsSequence$states=="E"])
```

The Viterbi algorithm

Note

Apart from the backtracking step, the Viterbi algorithm is similar to the forward algorithm. The key difference is the maximization in the former instead of summation in the latter.

The Baum-Welch algorithm

(maximum likelihood)

(sketch only)

The Baum-Welch algorithm

Intuitively, one may combine:
→ efficient likelihood evaluation,
→ (probabilistic) recovery of underlying state, and
→ ML estimation of Markov chain (lecture 1),
to conceive an algorithm along the following lines:

0) Initiate parameters **P**, **B** and **π**.
1) Apply Viterbi: most likely underlying sequence.
2) Given sequence, update estimates of **P**, **B** and **π**.
3) Iterate between 1) and 2) until convergence.

Baum-Welch does something similar, but formalized to ensure convergence to the maximum likelihood estimator.

Example

Sequence alignment

Example: sequence alignment

Consider two DNA sequences of two mammals:

human: CTATACG
mouse: CGATCG

In order to reconstruct the sequence in their common ancestor, the two sequences are aligned.

An alignment can be thought of as the result of sequential executions of the following operations:

- *match*,
- *substitution*,
- *insertion*, and
- *deletion*.

Example: sequence alignment

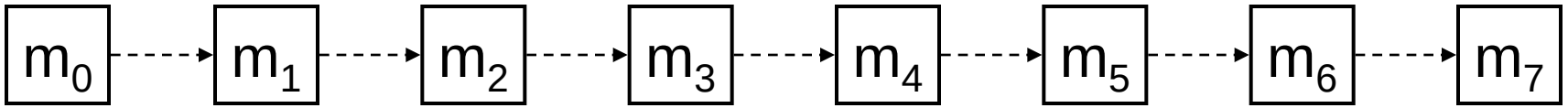
Hidden Markov models aid in alignment.

Formulate the alignment process in terms of a HMM. The operations *match*, *deletion*, and *insertion* are hidden states.

Suppose the two sequences match perfectly:

human: CTATACG
mouse: CTATACG (really: CGATCG)

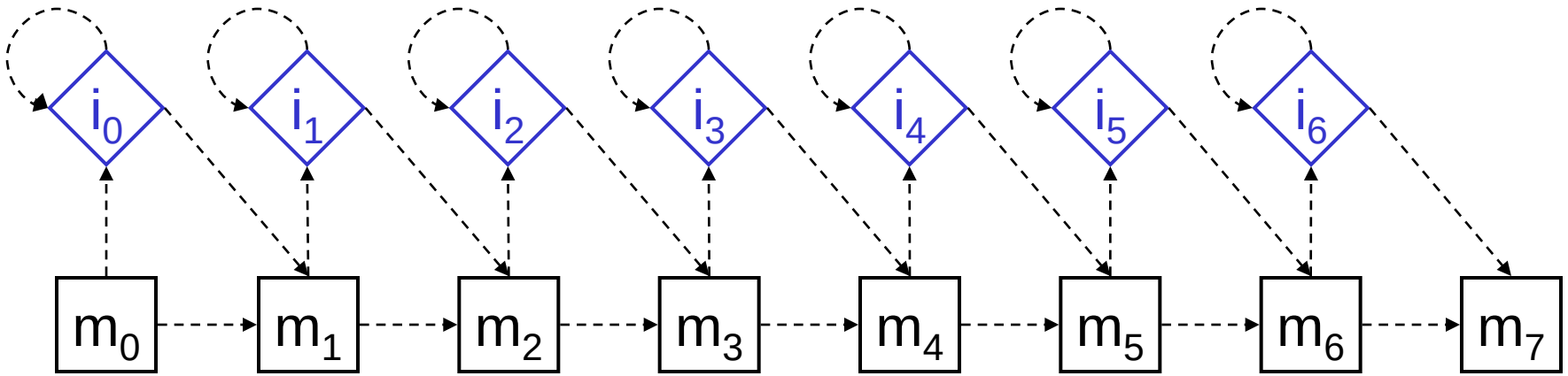
Only one hidden state is needed:



Example: sequence alignment

As the two sequences do not match perfectly, the HMM is extended with the insertion state:

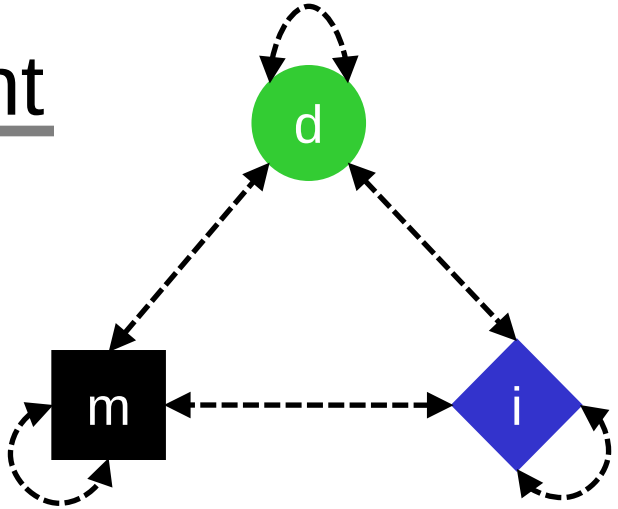
human: CTATACG
mouse: CGATCG -> CGTATACG

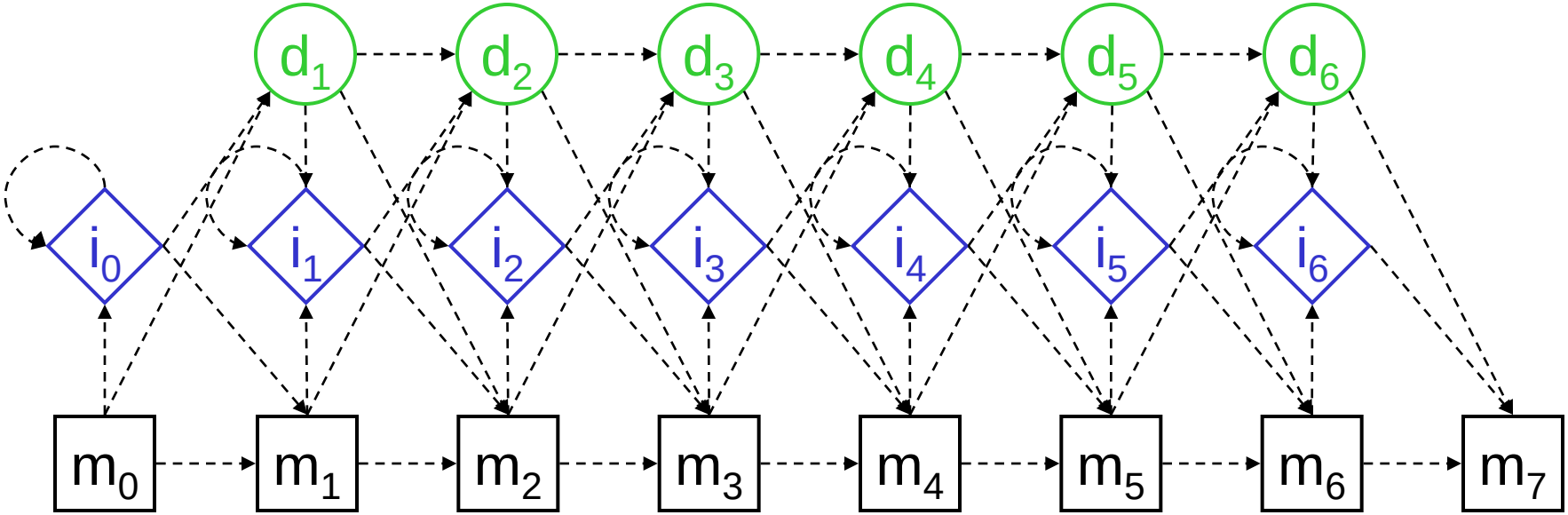


Example: sequence alignment

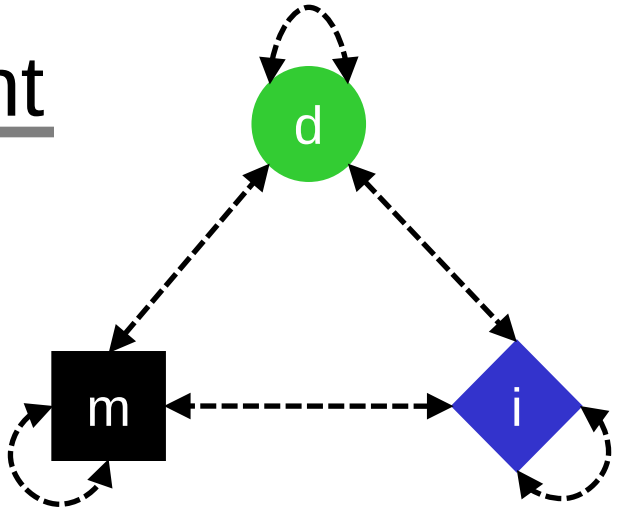
Now add the deletion state:

human : CTATACG
mouse : CGTATACG
-> CGTATACG





Example: sequence alignment



Remarks:

1) Deletion state does not emit: affects the dimension of the emission matrix.

2) Consequently, the underlying sequence may be longer than the observed.

***t*:**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
human:	C		T	A	T	A	C	G
underlying:	m	d	i	m	m	i	m	m
mouse:	C	G		A	T		C	G

3) First hidden state is defined as observed start state.

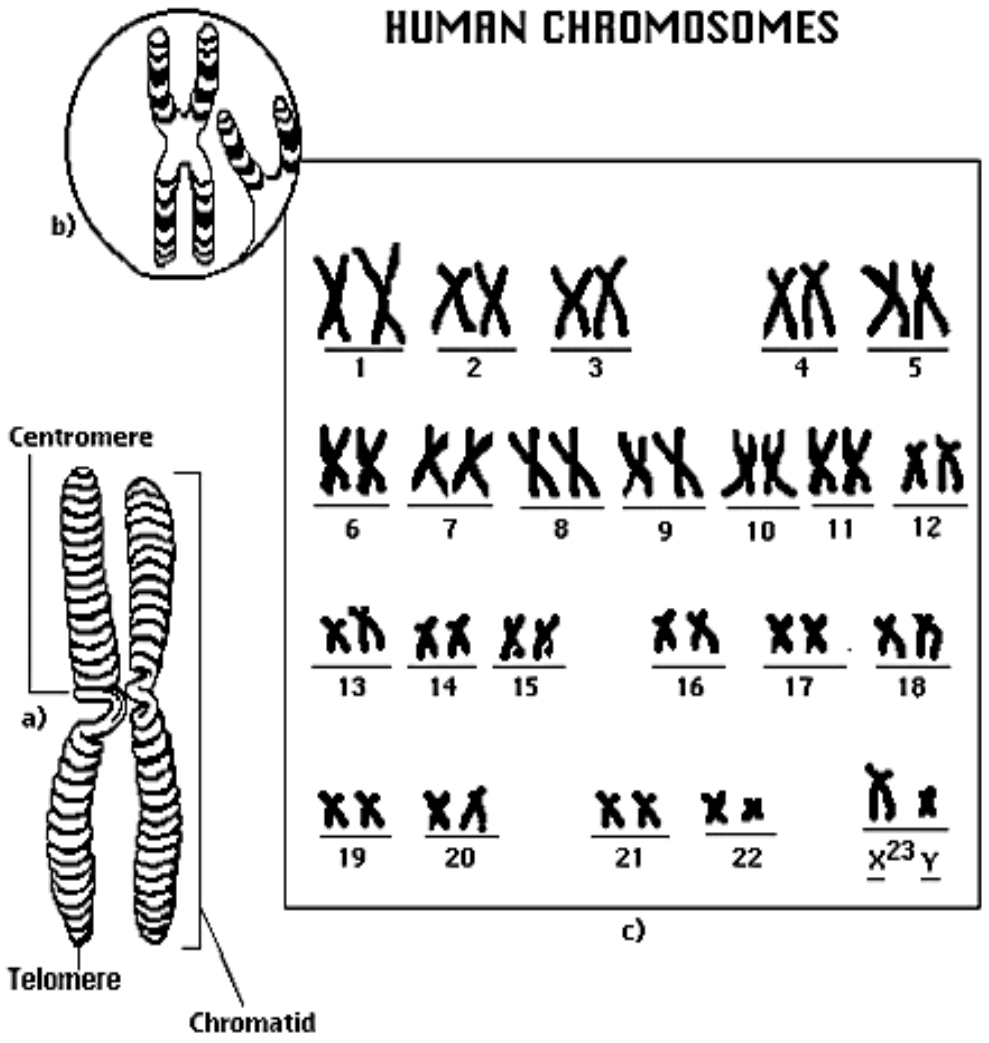
Example

Array CGH

Example: array CGH

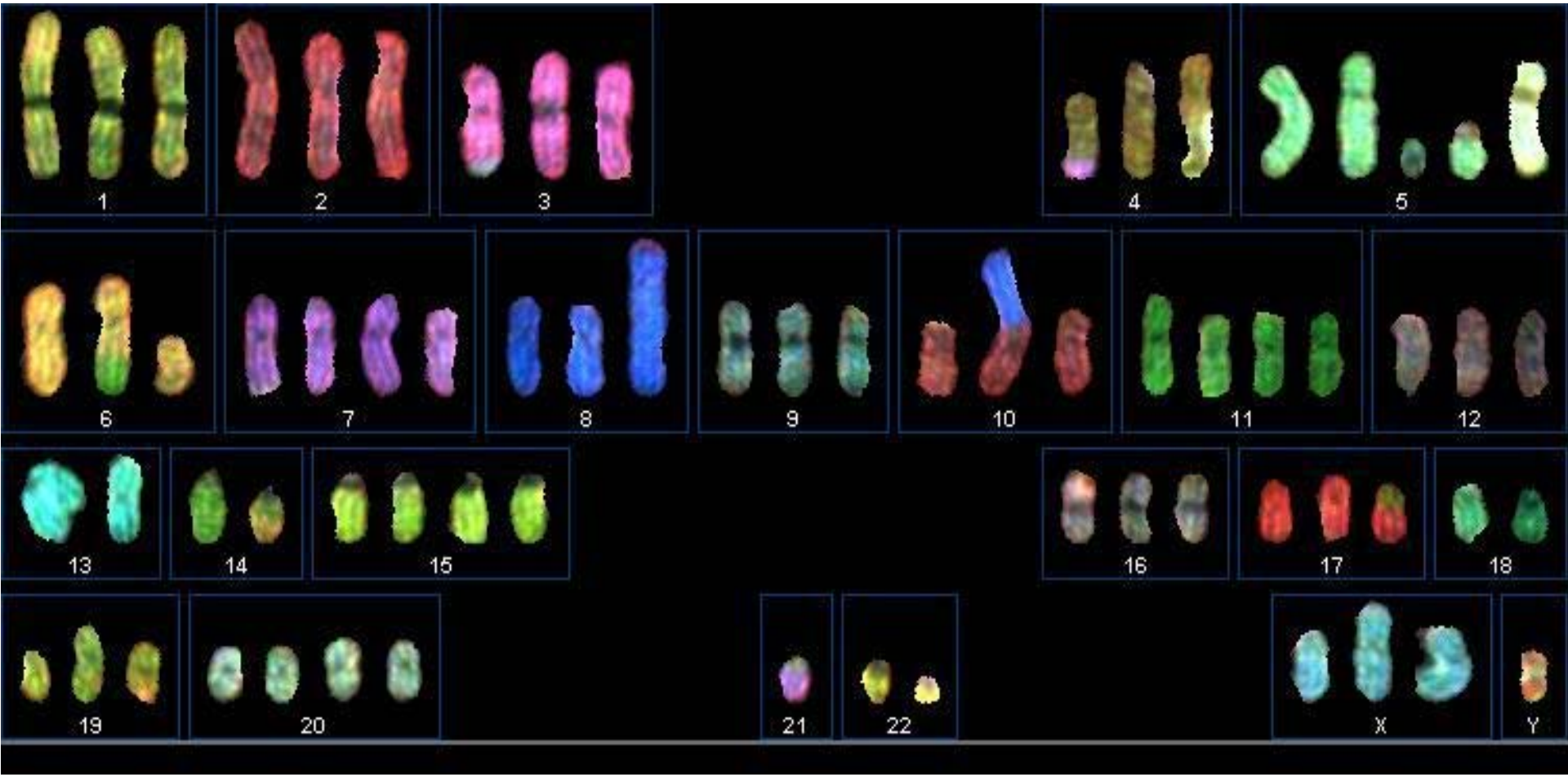
DNA copy number of a genomic segment is simply the number of copies of that segment present in the cell under study.

Healthy normal cell:
chr 1 : 2
…
chr 22 : 2
chr X : 1 or 2
chr Y : 0 or 1



Example: array CGH

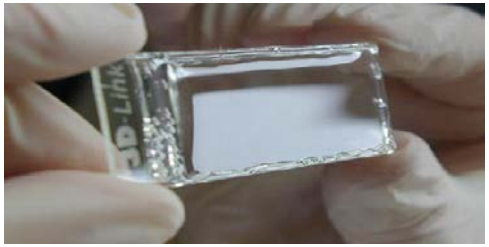
Chromosomes of a tumor cell



Technique: SKY

Example: array CGH

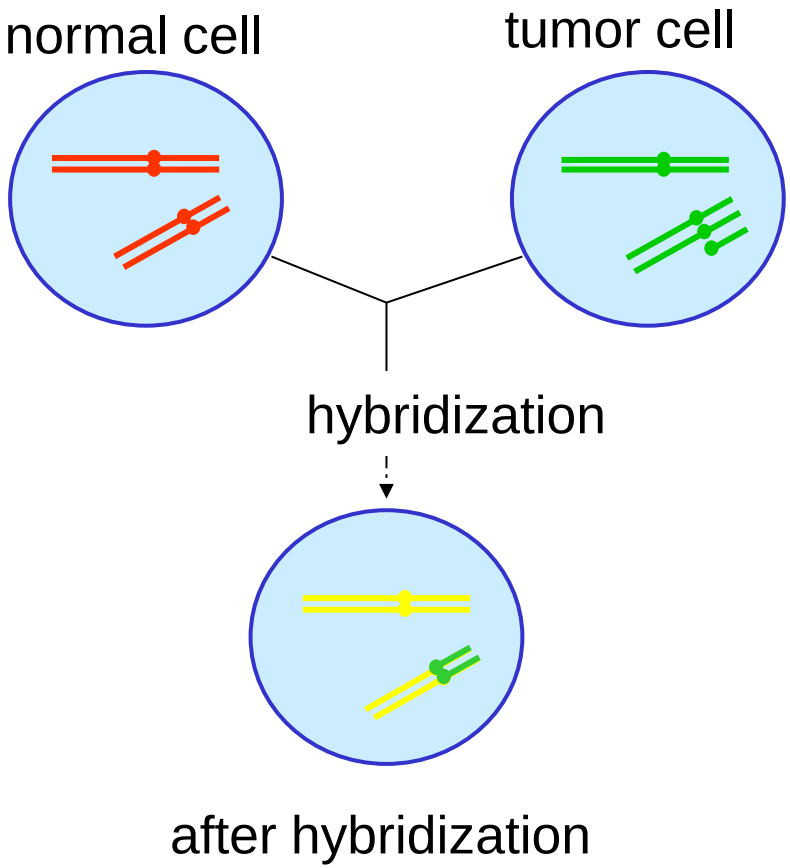
DNA copy number may be measured genome-wide in high-throughput fashion using microarrays.



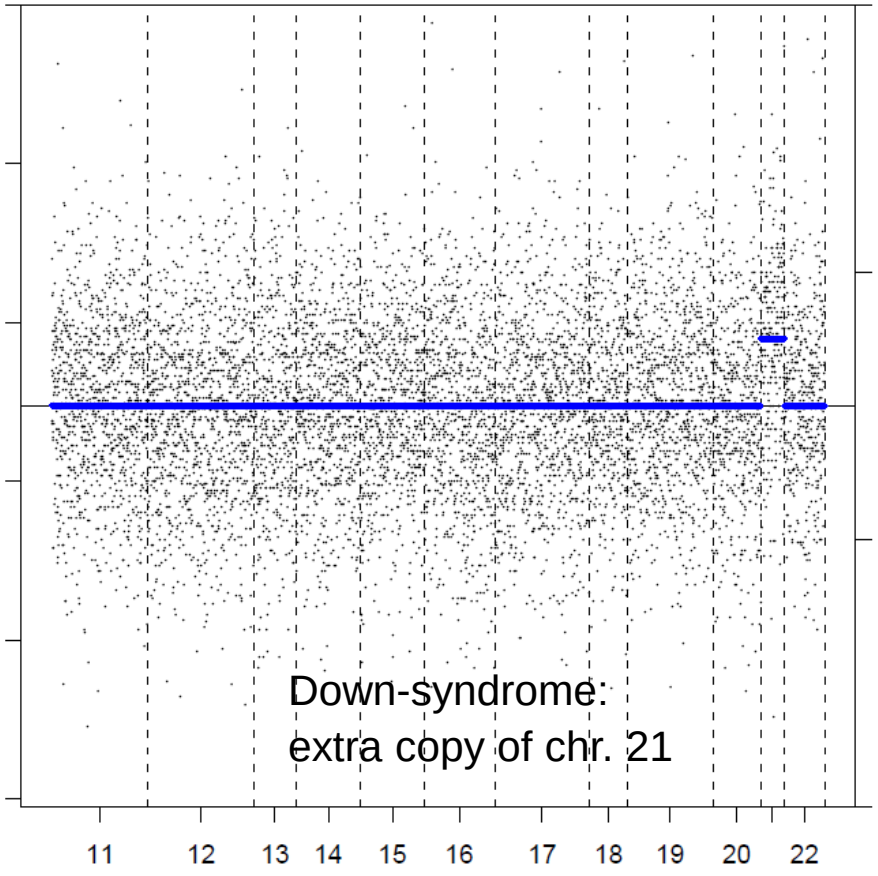
- Test and reference (assumed to have DNA copy number 2) samples are labeled differently and hybridized together.
- Under ideal circumstances, the intensity of an array element is linearly proportional to the abundance of the corresponding DNA sequence.
- Log_2 ratios of test and reference intensities reflect the relative copy number in the test sample compared to that in the reference sample.

Example: array CGH

Principle of array CGH.

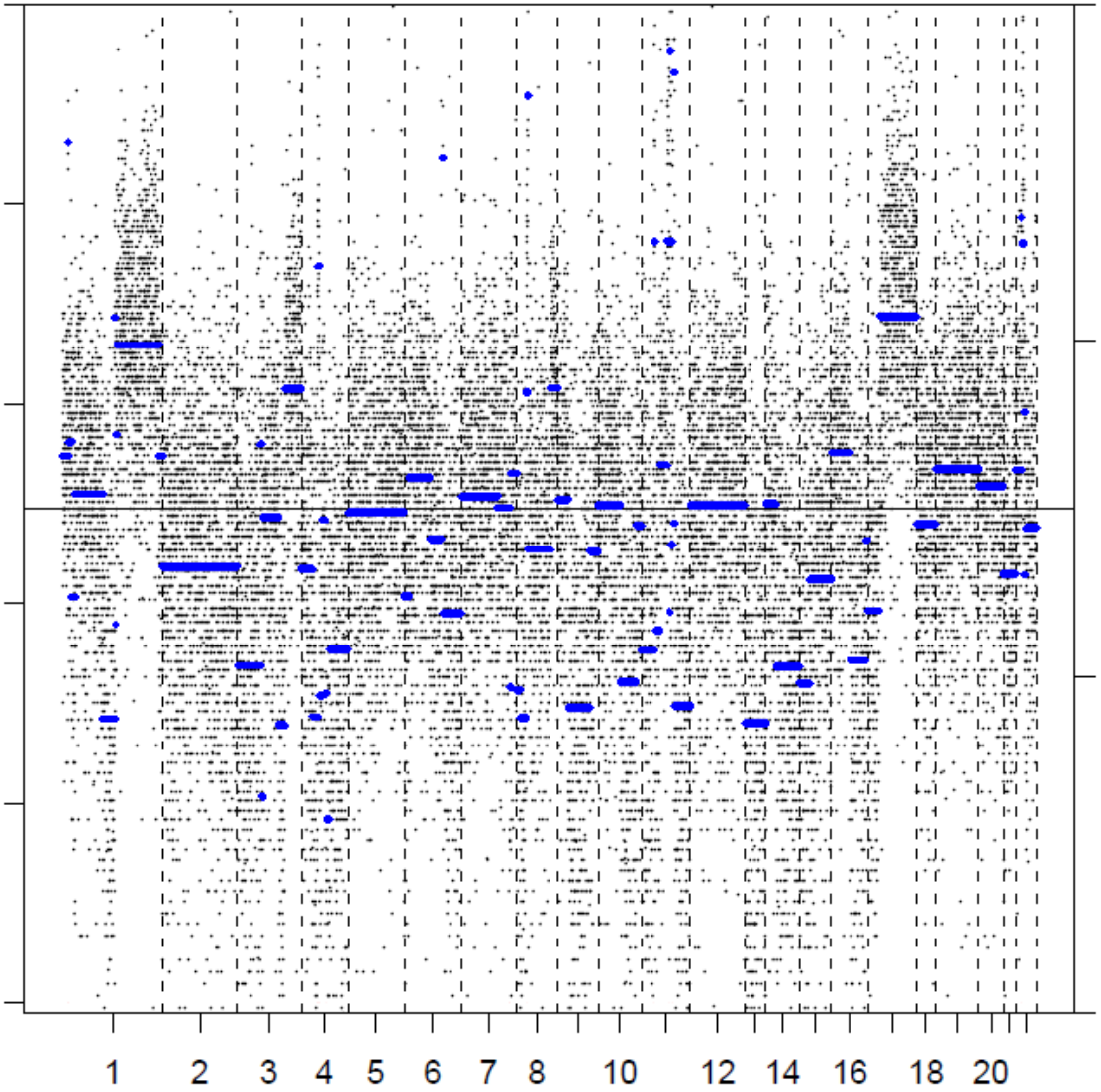


Copy numbers of each genomic segment are assessed simultaneously.



Example: array CGH

DNA copy number profile of a cancer sample



Example: array CGH

Problem: determine the aberrated genomic segments from the observed $\log_2$ ratios.

Solution: model the DNA copy number profile by a HMM.

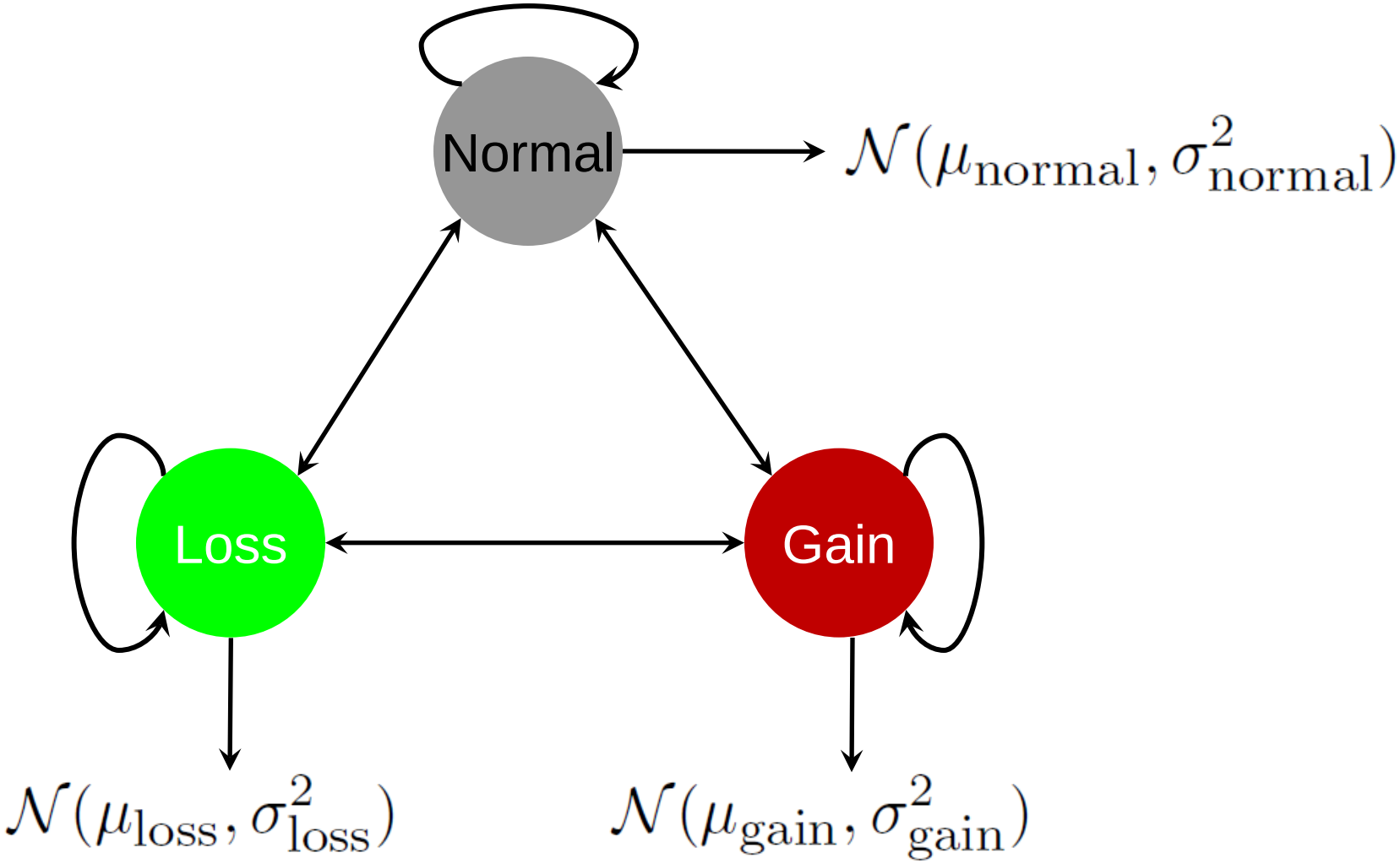
Ingredients for the HMM: three hidden states:

- L : *loss* : < 2 copies
- N : *normal* : 2 copies
- G : *gain* : > 2 copies

Given the hidden states, a *continuous* value is emitted.

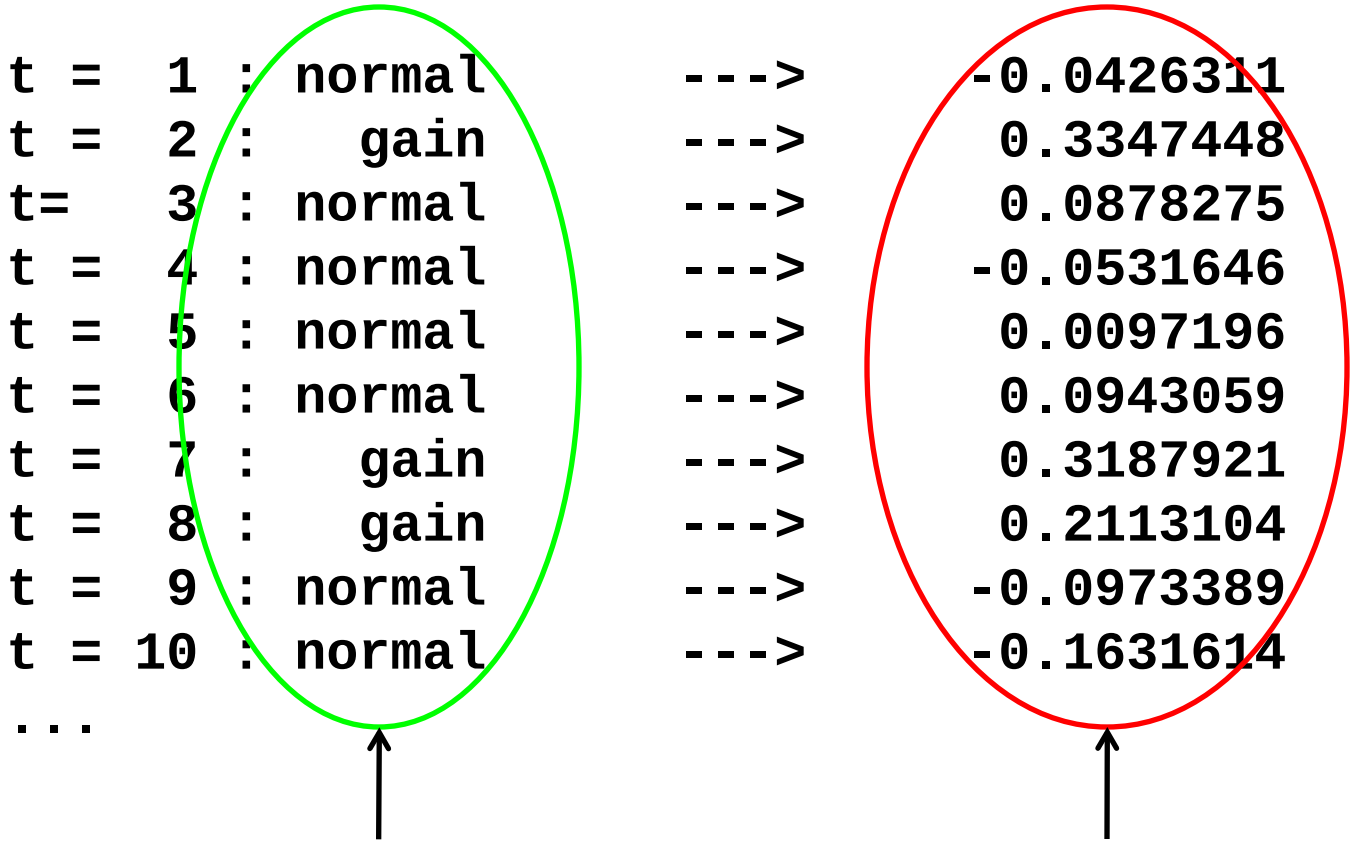
Example: array CGH

HMM architecture for array CGH



Example: array CGH

Sampling from this HMM:

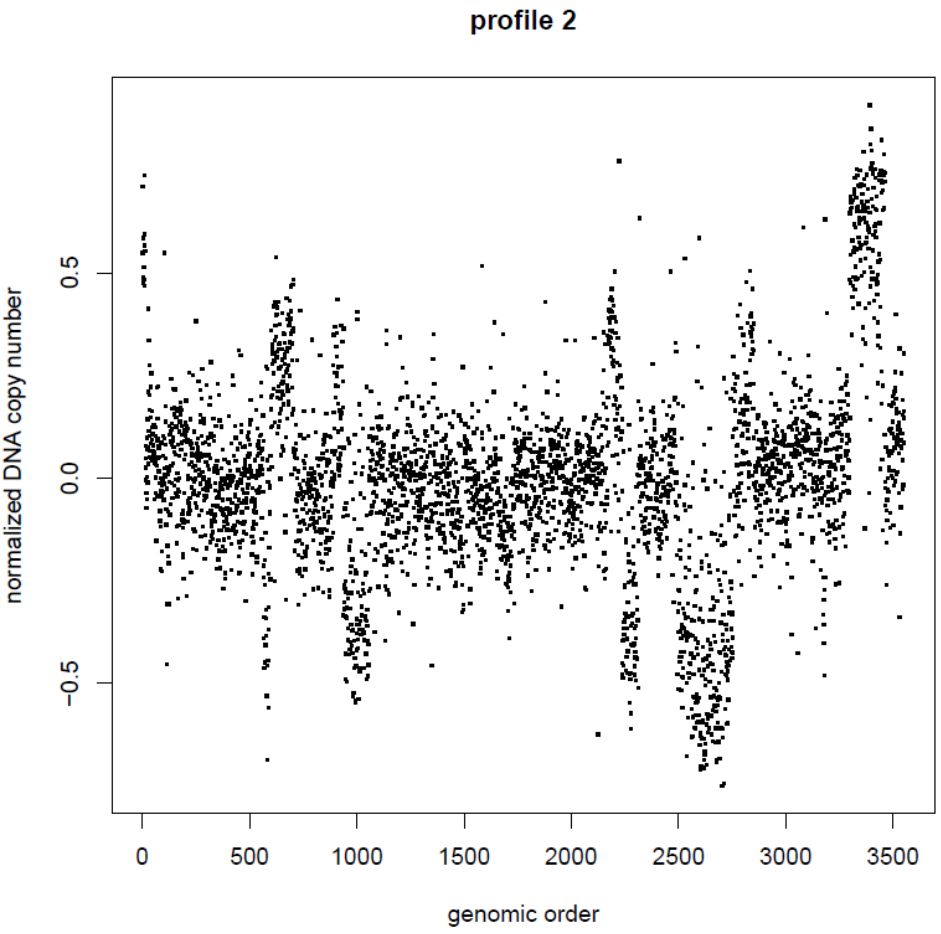


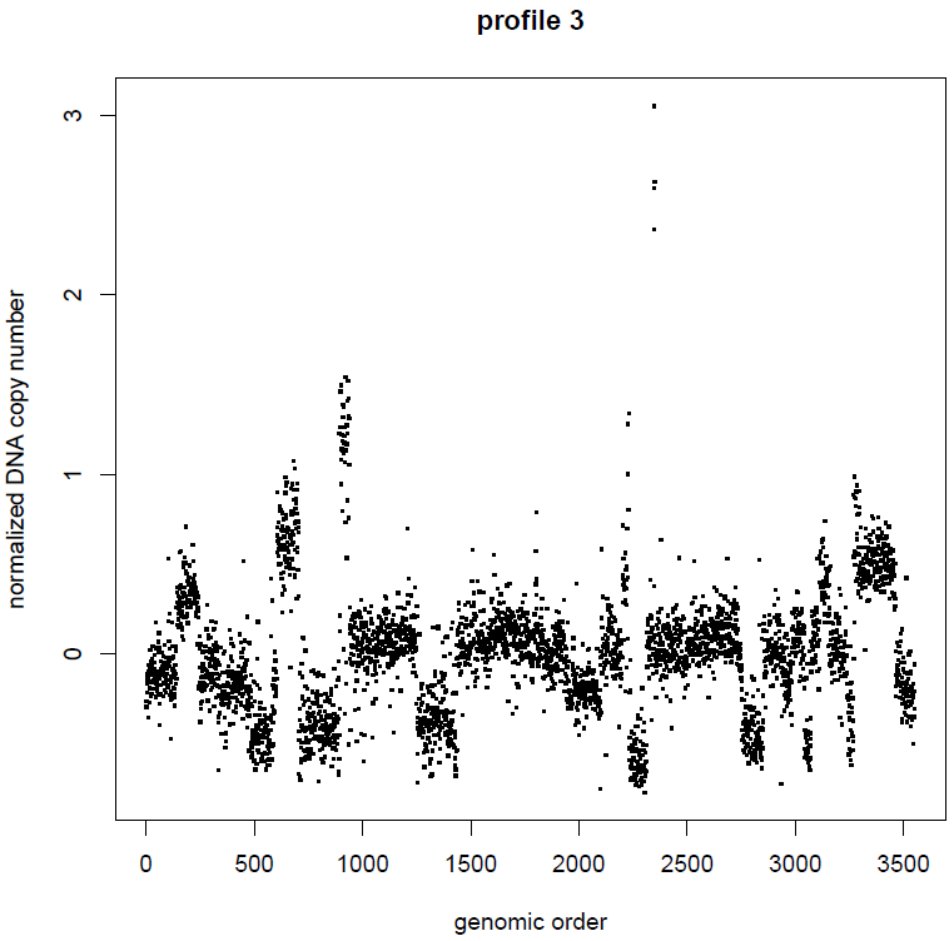
unobserved sequence of states

observed sequence of $\log_2$ ratios

Example: array CGH

DNA copy number profiles of two cervix cancer samples.





Example: array CGH

Analysis flow

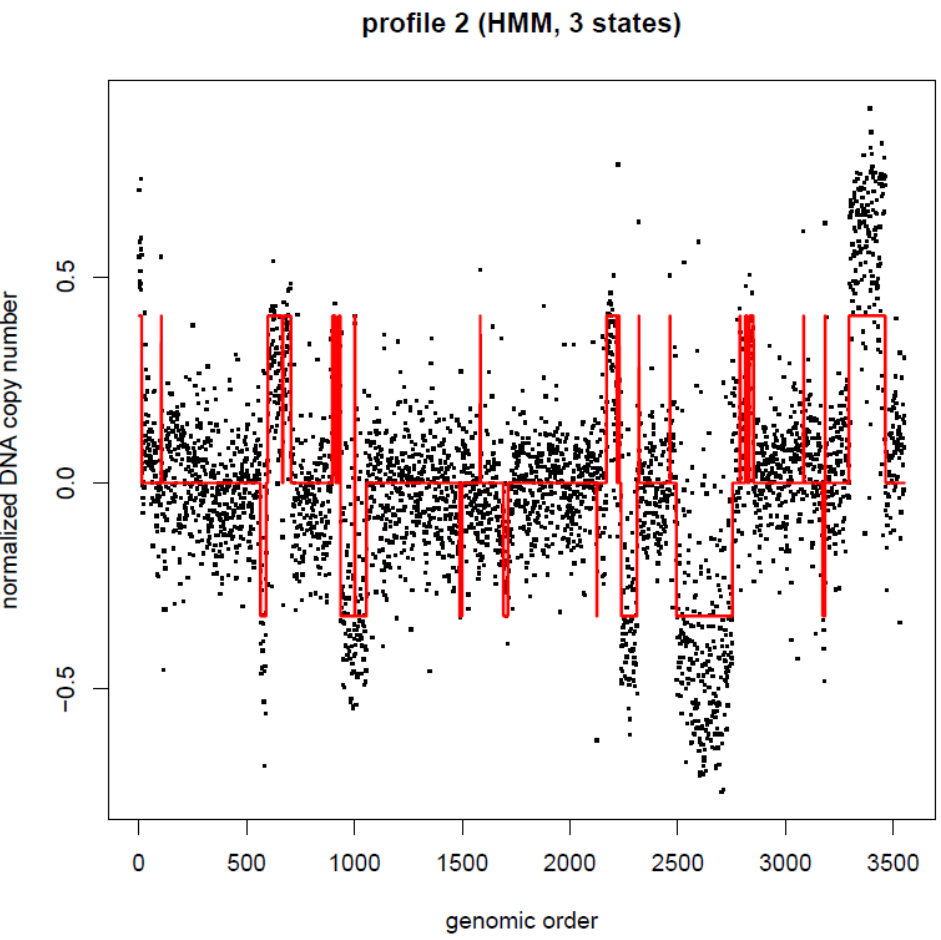
- Fit HMM to data by means of ML (Baum-Welch).

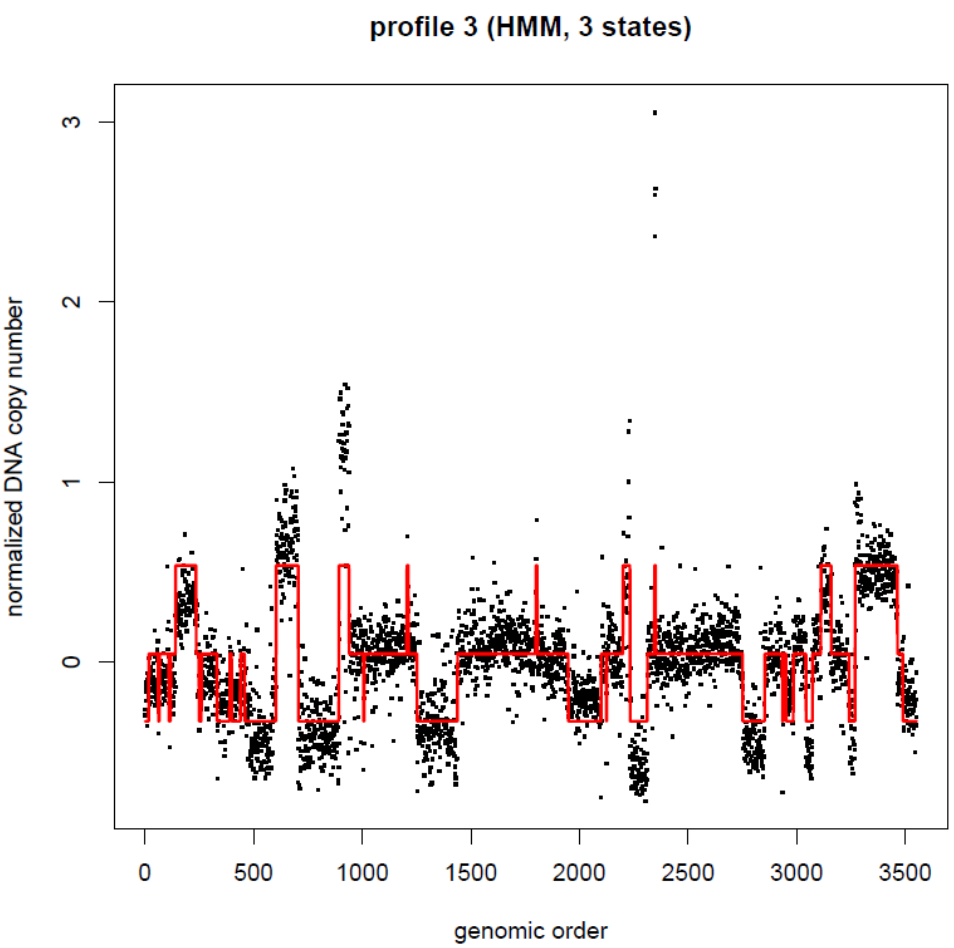
```
> summary(hmmFit)
$delta
[1] 1.000000e+00 1.801274e-45 2.257732e-58

$Pi
             [,1]       [,2]        [,3]
[1,] 0.979794054 0.01695689 0.003249055
[2,] 0.011131489 0.98481411 0.004054398
[3,] 0.005139211 0.01587148 0.978989311

$pm
$pm$mean
[1] -0.32802689  0.04630948  0.53721098

$pm$sd
[1] 0.1850411 0.1335572 0.3447424
```

- Given ML estimates, calculate hidden states (Viterbi)

Example: array CGH

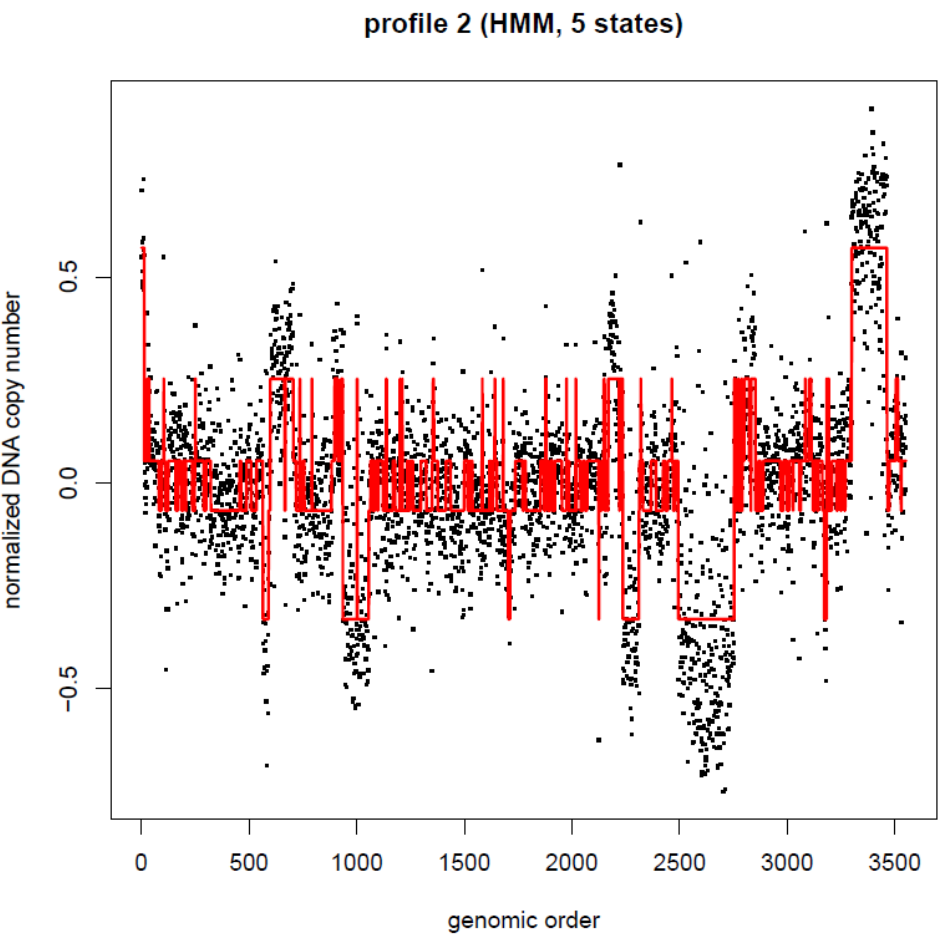
HMM result with three states.

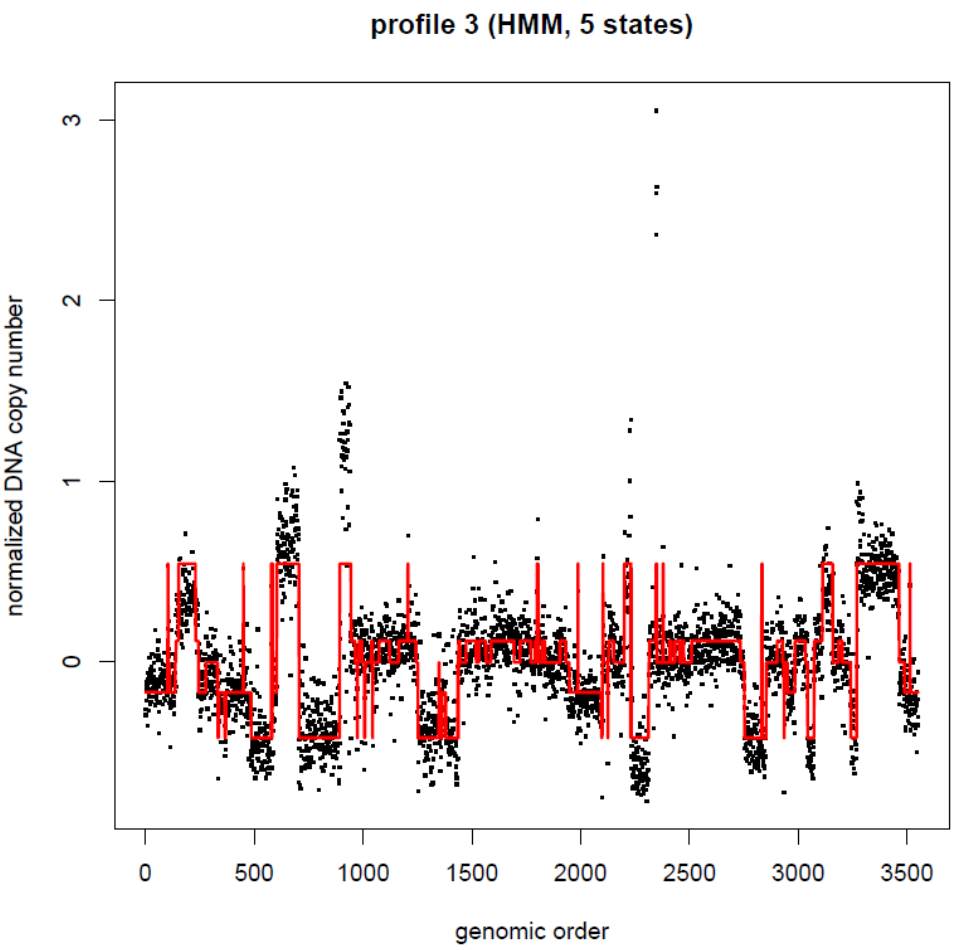




Example: array CGH

HMM result with five states.





Example: array CGH

In R:

```
> # activate libraries
> library(CGHcall)
> library(HiddenMarkov)

> # load data
> data(Wilting)

> # specify profile to analyze
> profNo <- 3

> # put in particular format
> cgh <- make_cghRaw(Wilting)

> # first preprocess the data
> rawCGH <- preprocess(cgh)

> # normalize the data (global median normalization)
> normCGH <- normalize(rawCGH)
> plot(copynumber(normCGH)[,profNo], pch=".", cex=2)
```

Example: array CGH

R code (continued)

```
> # create object for HMM-fitting
> x <- dthmm(copynumber(normCGH)[,profNo],
        Pi=matrix(c(.8,.1,.1,.1,.8,.1,.1,.1,.8),3),
        delta=c(0.1, 0.8, 0.1), "norm",
        pm=list(mean=c(-0.2, 0, 0.2), sd=c(0.1, 0.1, 0.1)))

> # fit HMM
> hmmFit <- BaumWelch(x)

> # estimate hidden states
> states <- Viterbi(hmmFit)

> # overlay data with estimated hidden states
> stateValues <- hmmFit$pm$mean
> plot(copynumber(normCGH)[, profNo], pch=".", cex=2)
> lines(stateValues[states], col="red", lwd=2)
```

Example: array CGH

Note

The HMM is fitted to data from the whole genome. Effectively, this assumes the chromosomes are glued together (ordered by their numbers). This is of course nonsense.

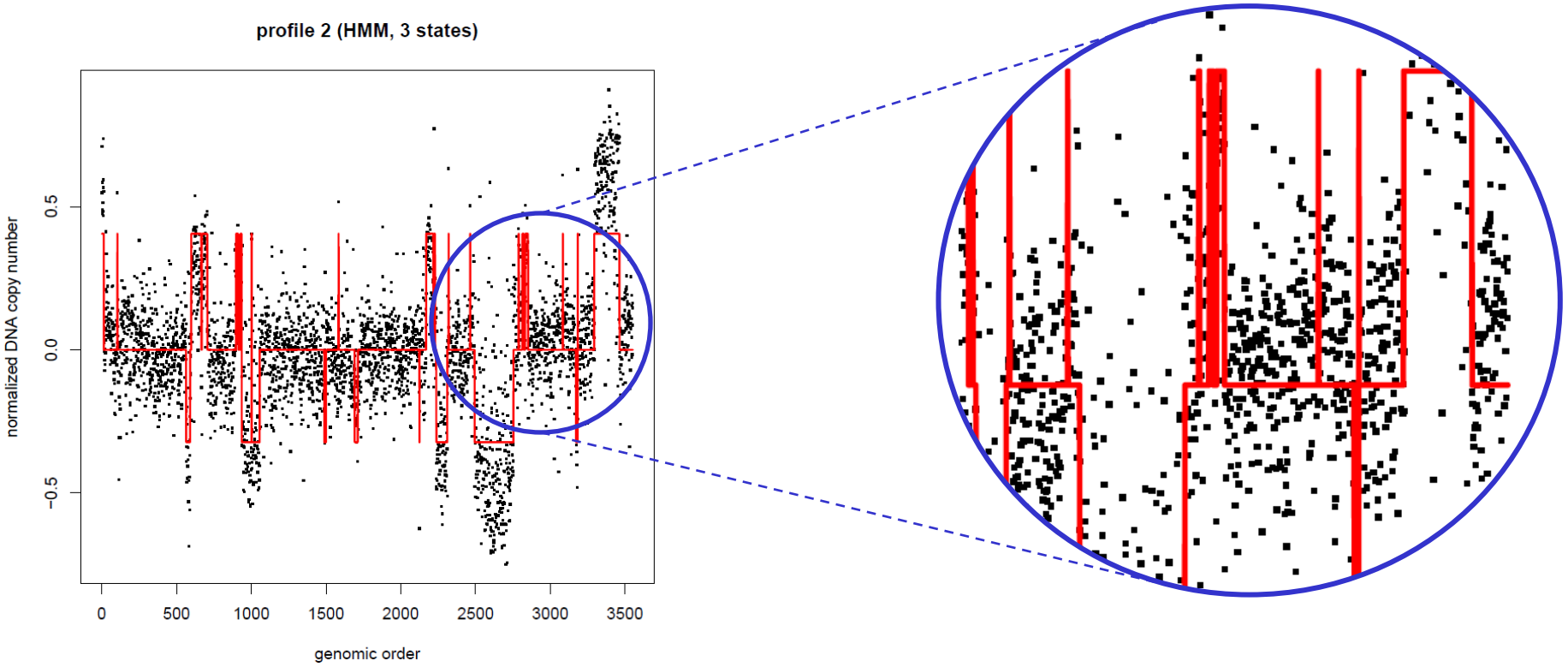
Exercise

- Modify the code to fit the HMM to each chromosome separate, and overlay the data with the resulting fit.
- Moreover, also investigate the number of states to be used.

Hidden semi-Markov model

Hidden semi-Markov model

The HMM analysis suggest that within only a few steps multiple hidden states have been visited. This may be unrealistic biologically.



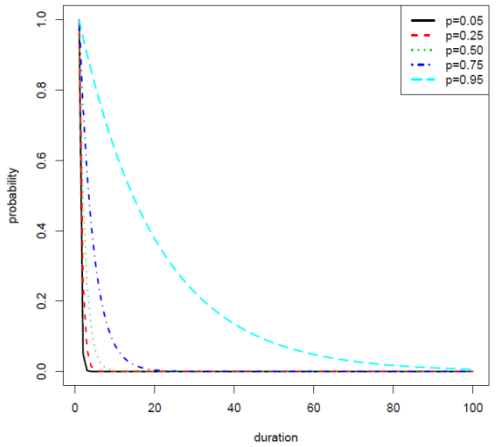
Hidden semi-Markov model

Consider the intron-exon charicature of the DNA again.

How long do we spend in a particular state:

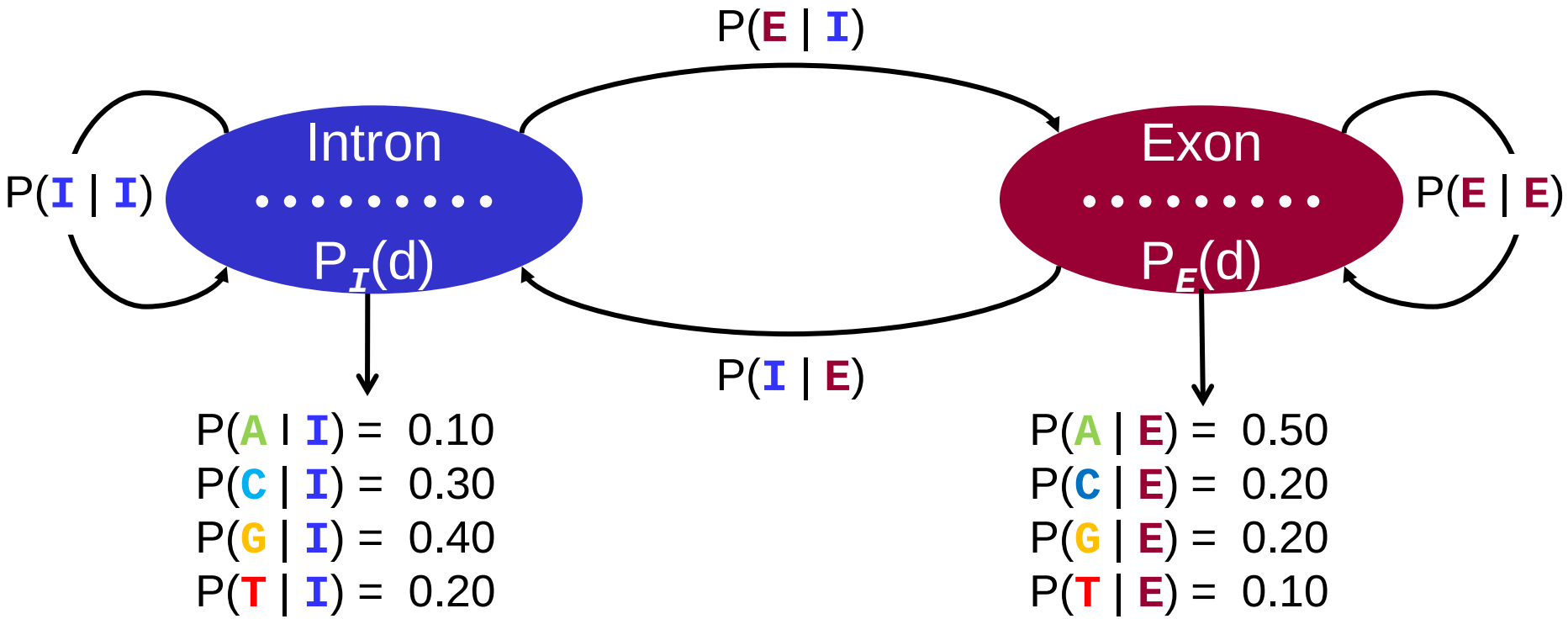
$$
\begin{aligned}
& P(d \text{ time steps time spent in state } k) \\
&= P(X_t \neq k, X_{t+1} = k, \ldots, X_{t+d} = k, X_{t+d+1} \neq k) \\
&= P(X_t \neq k) \, (\mathbf{P})_{\neg k,k} \, (\mathbf{P})_{kk}^{d-1} \, (\mathbf{P})_{k,\neg k}
\end{aligned}
$$

Short durations more likely than long durations.



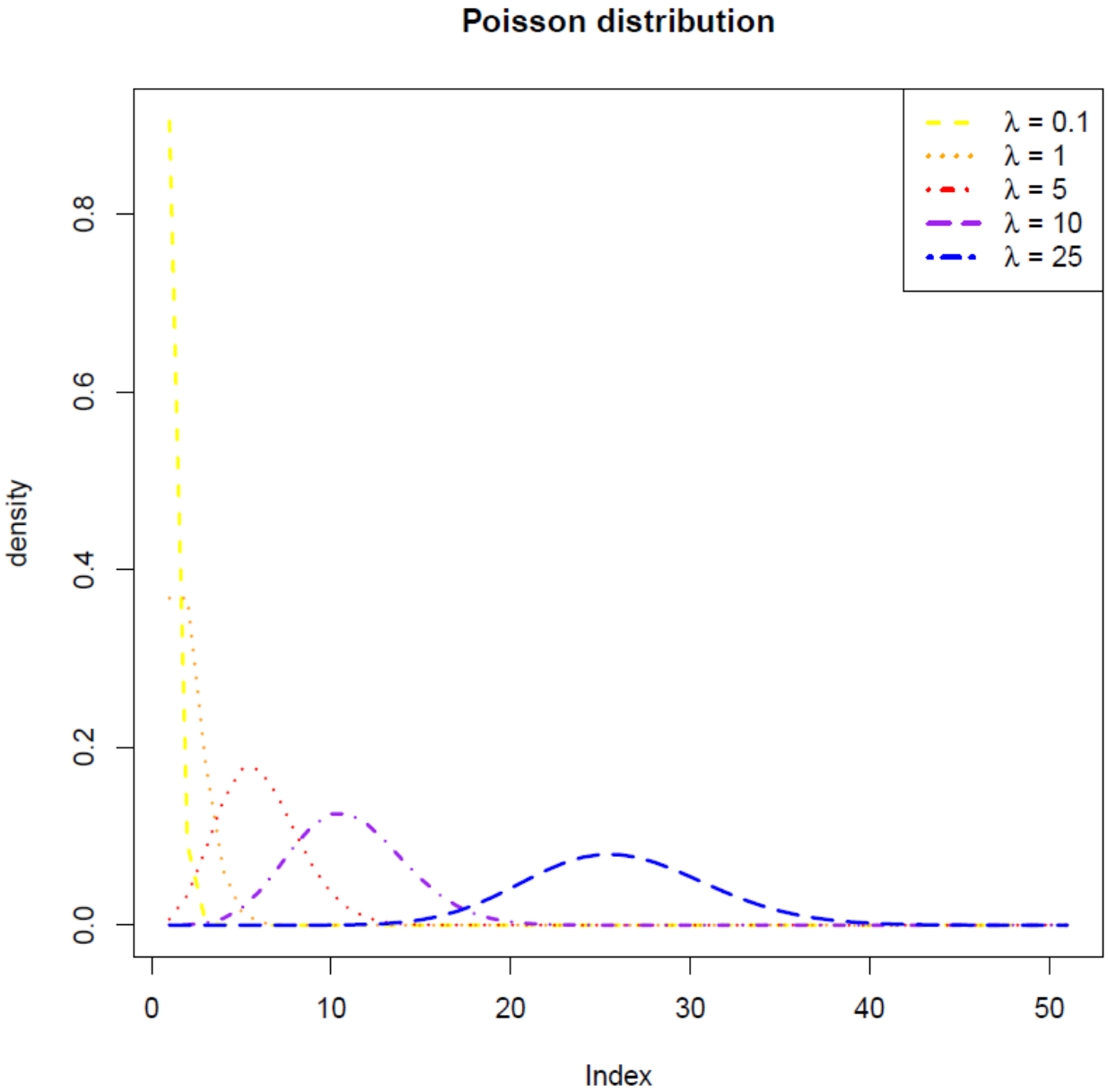
Hidden semi-Markov model

Modify the HMM to allow more realistic duration times.



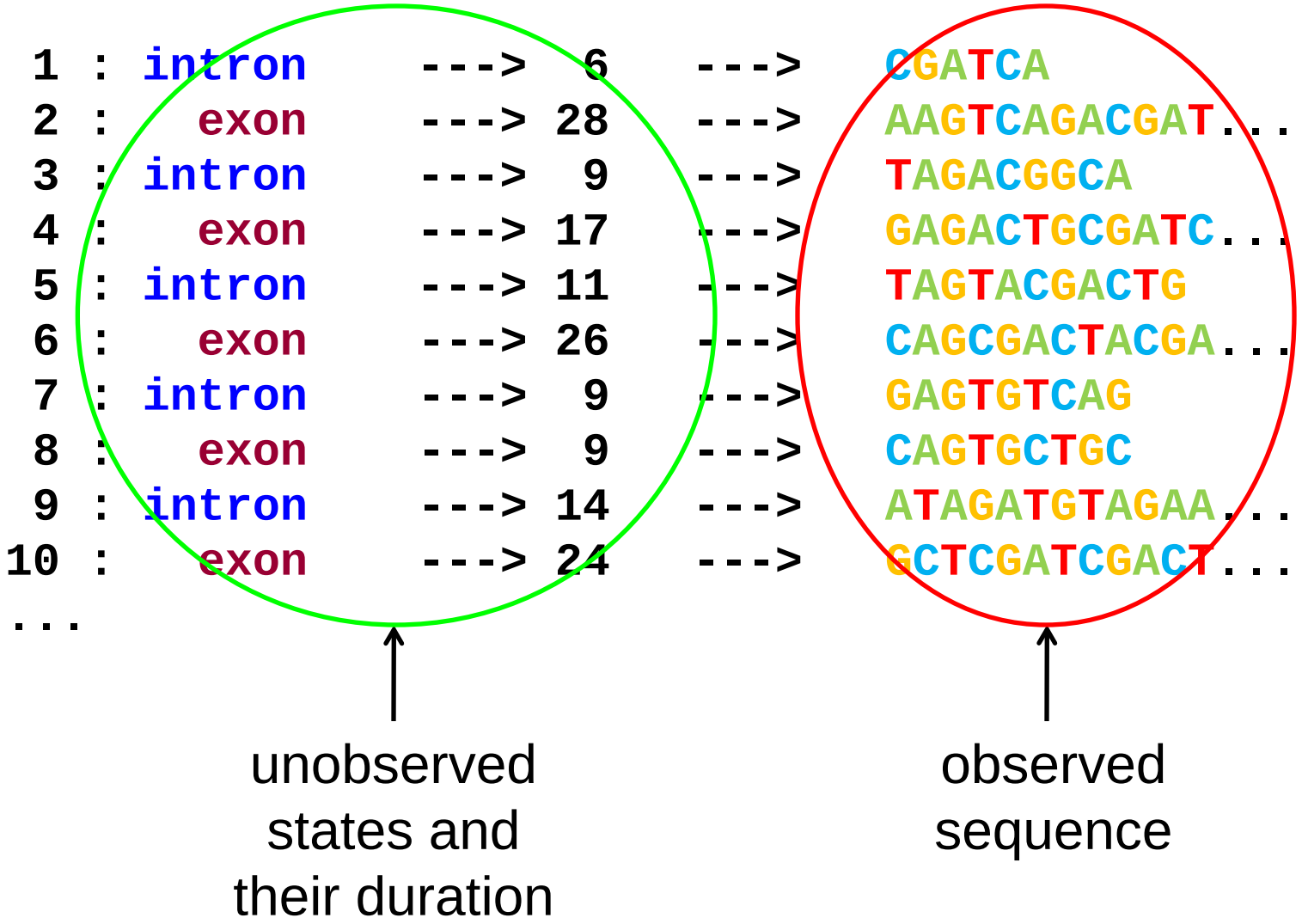
where $P_I(d)$ and $P_E(d)$ are, e.g. Poisson distributions describing the duration time.

Hidden semi-Markov model



Hidden semi-Markov model

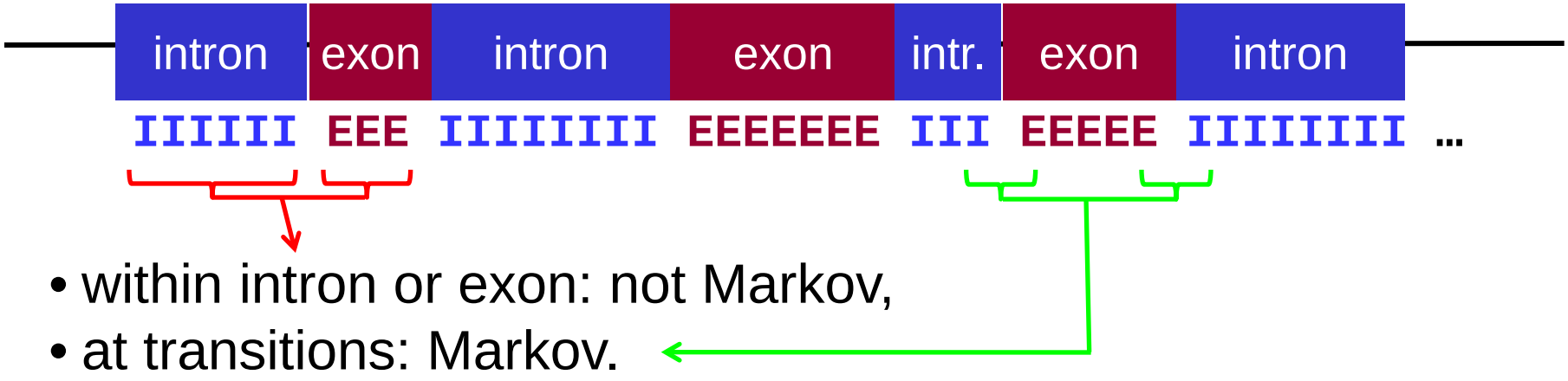
Sampling from this modified HMM:



Hidden semi-Markov model

The modified HMM is called a *hidden semi-Markov model*.

Within this model the Markov property holds for state transitions, but no longer for sequential observations.

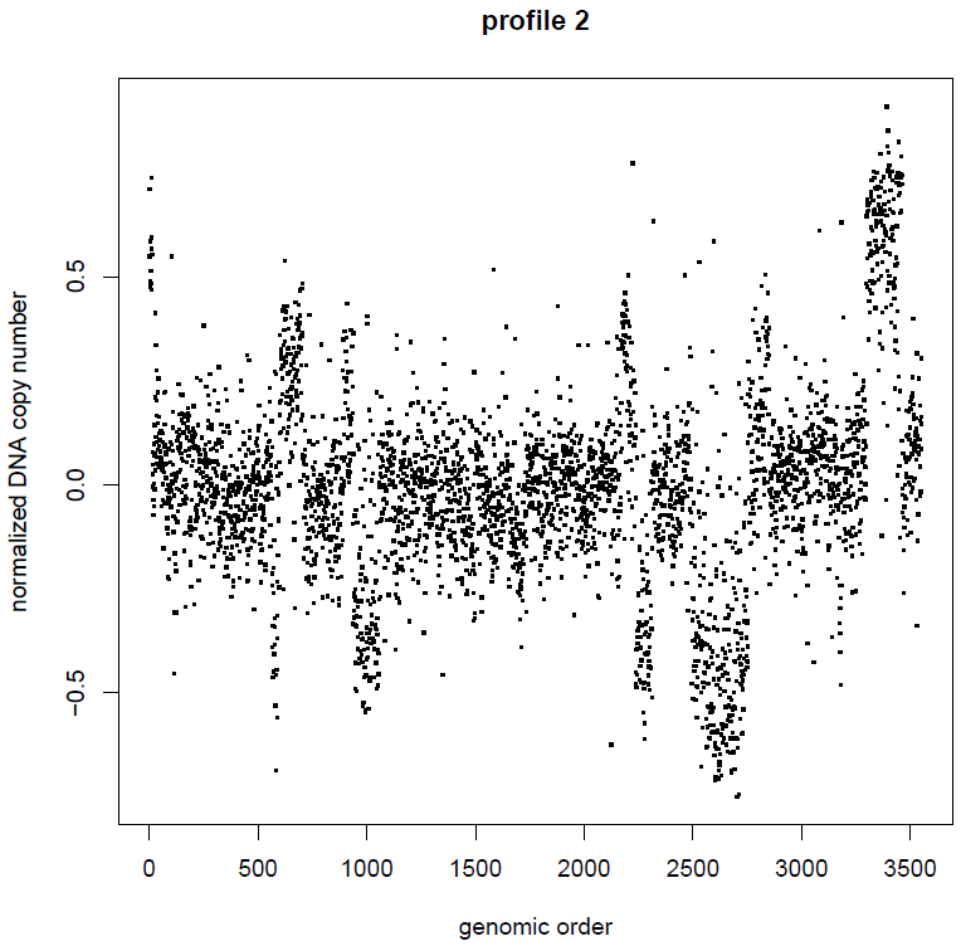


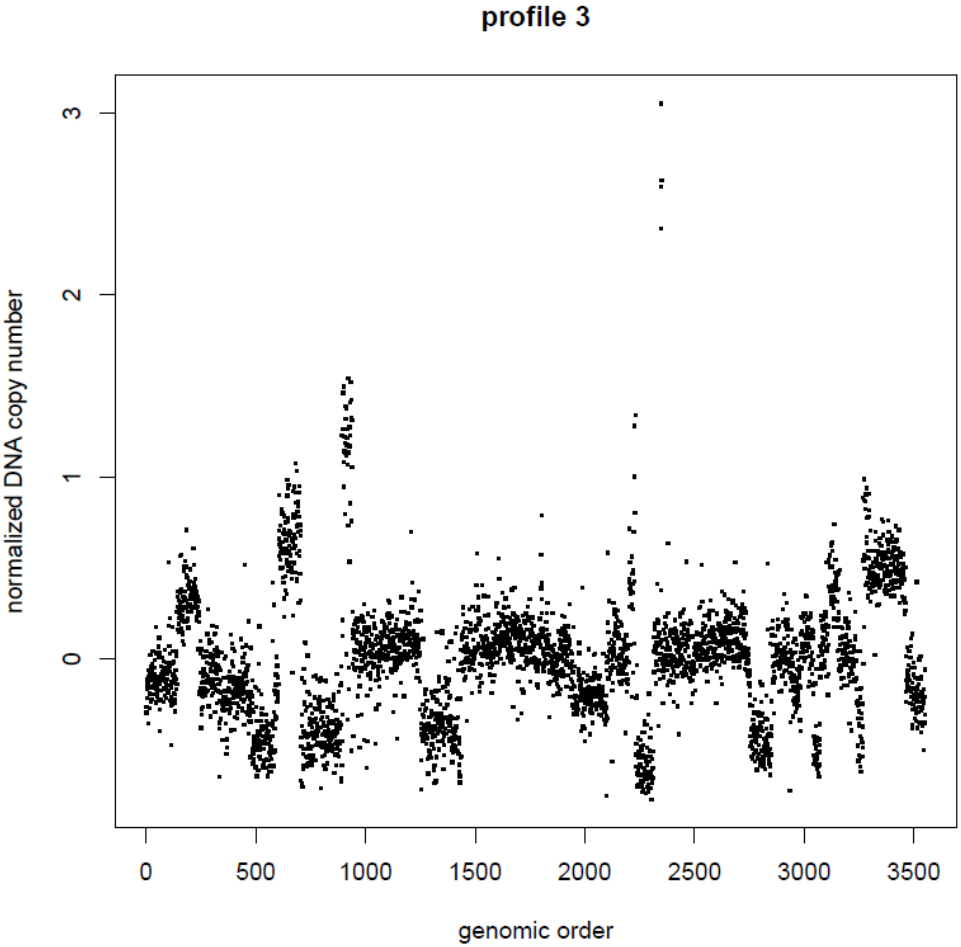
Note: with two states, the unobserved sequence of states is deterministic, alternating between the two states.

Example: array CGH (revisited)

Example: array CGH (revisited)

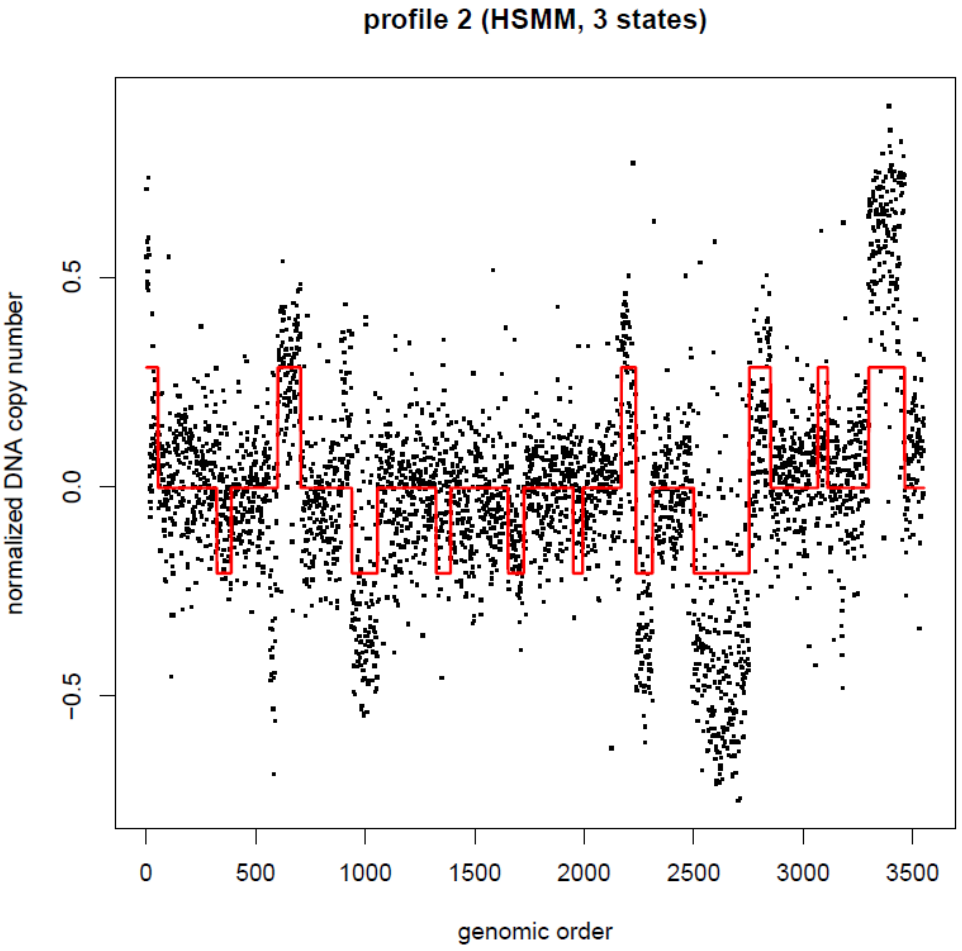
Recall: array CGH measures DNA copy number in high - throughput fashion. Below two cervix cancer profiles.

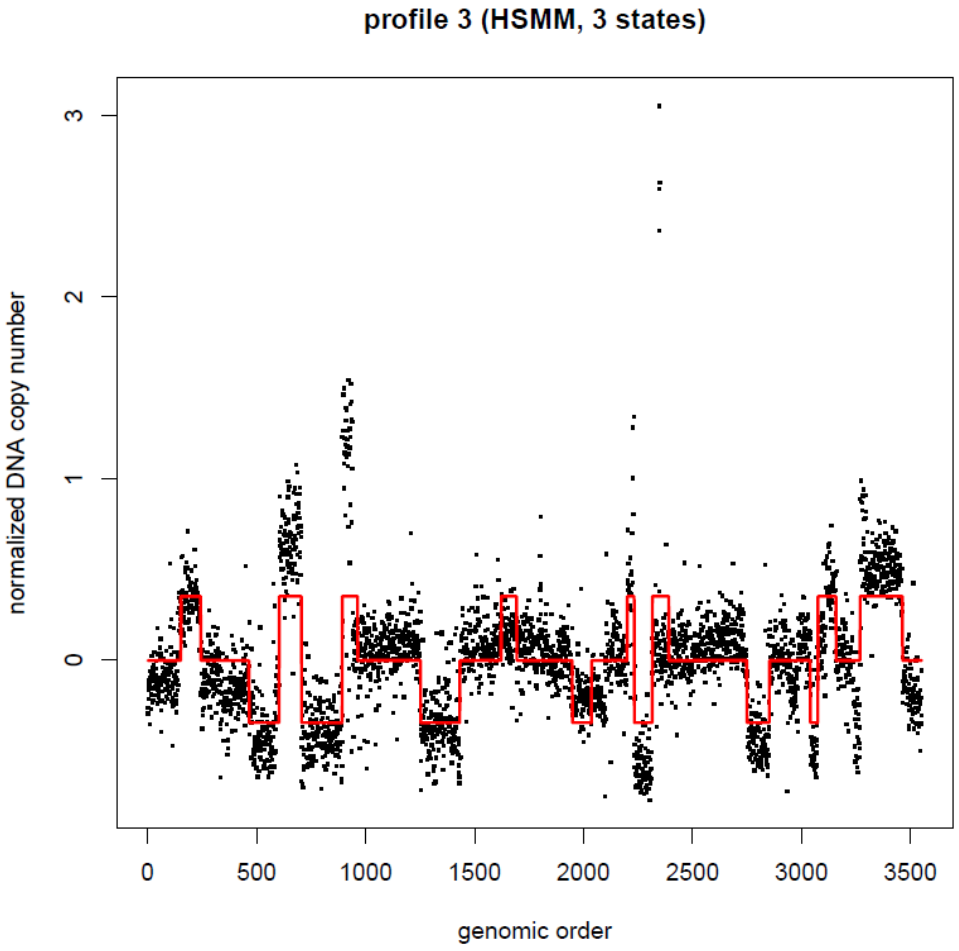




Example: array CGH (revisited)

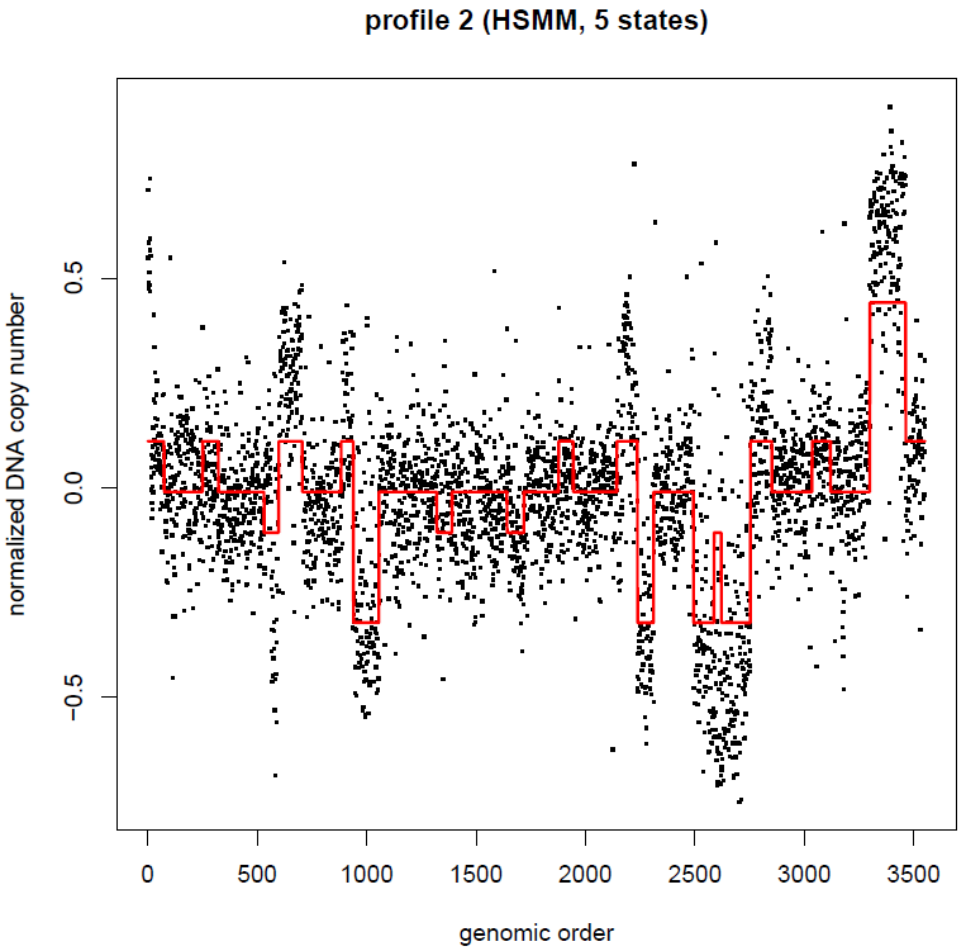
HSMM result with three states.

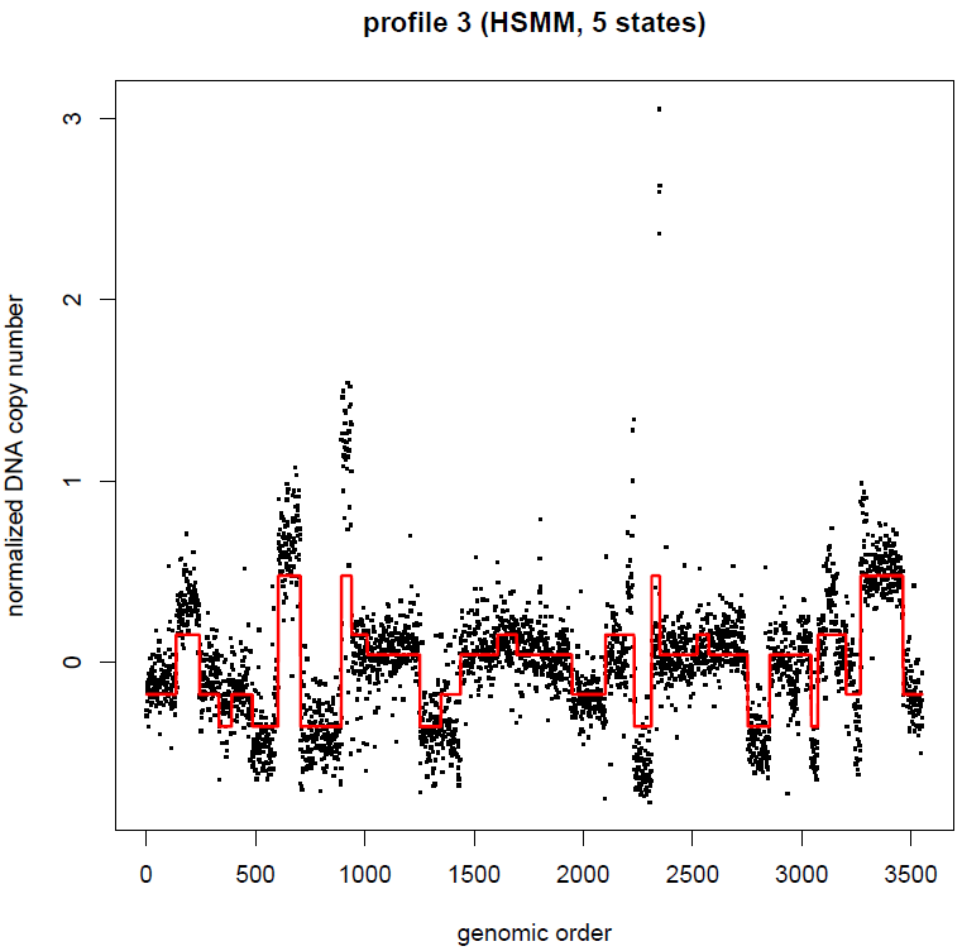




Example: array CGH (revisited)

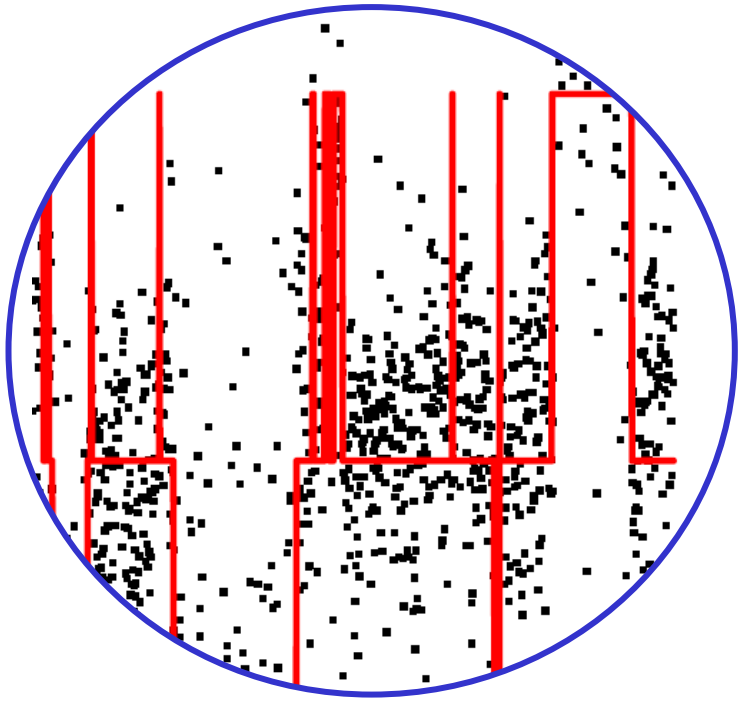
HSMM result with five states.

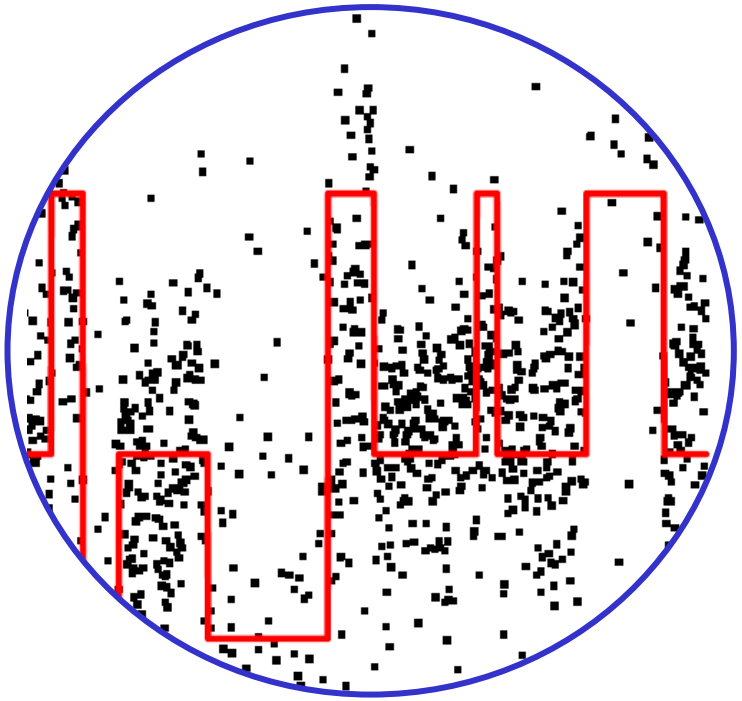




Example: array CGH (revisited)

The HSMM analysis yields longer duration times than the HMM for the DNA copy number profile of the cervix cancer samples.





Supplementary Material:
Proofs of recursive
relations of the
forward and Viterbi variable

The forward algorithm

$$
\begin{aligned}
\alpha_t(i) &= P(Y_1, \ldots, Y_t, X_t = S_i \,|\, \boldsymbol{\lambda}) \\
&= \sum_{j=1}^{N} P(Y_1, \ldots, Y_t, X_t = S_i, X_{t-1} = S_j \,|\, \boldsymbol{\lambda}) \\
&= \sum_{j=1}^{N} P(Y_t, X_t = S_i \,|\, Y_1, \ldots, Y_{t-1}, X_{t-1} = S_j; \boldsymbol{\lambda}) \\
&\qquad \times P(Y_1, \ldots, Y_{t-1}, X_{t-1} = S_j \,|\, \boldsymbol{\lambda}) \\
&= \sum_{j=1}^{N} P(Y_t \,|\, X_t = S_i; \boldsymbol{\lambda})\, P(X_t = S_i \,|\, X_{t-1} = S_j; \boldsymbol{\lambda}) \\
&\qquad \times P(Y_1, \ldots, Y_{t-1}, X_{t-1} = S_j \,|\, \boldsymbol{\lambda}) \\
&= b_i(Y_t) \sum_{j=1}^{N} \alpha_{t-1}(j)\, (\mathbf{P})_{ji}
\end{aligned}
$$

The Viterbi algorithm

For this variable too there exists a recursive relationship:

$$
\begin{aligned}
\delta_{t+1}(j) &= \max_{X_1,\ldots,X_t} P(X_1,\ldots,X_t,X_{t+1}=S_j,Y_1,\ldots,Y_{t+1}\,|\,\boldsymbol{\lambda}) \\
&= [\max_{X_1,\ldots,X_t} P(X_1,\ldots,X_t,Y_1,\ldots,Y_t\,|\,\boldsymbol{\lambda}) \\
&\qquad P(X_{t+1}=S_j\,|\,X_t,\boldsymbol{\lambda})] \\
&\qquad \times P(Y_{t+1}\,|\,X_{t+1}=S_j,\boldsymbol{\lambda}) \\
&= [\max_i \max_{X_1,\ldots,X_{t-1}} P(X_1,\ldots,X_t=S_i,Y_1,\ldots,Y_t\,|\,\boldsymbol{\lambda}) \\
&\qquad P(X_{t+1}=S_j\,|\,X_t=S_i,\boldsymbol{\lambda})] \\
&\qquad \times P(Y_{t+1}\,|\,X_{t+1}=S_j,\boldsymbol{\lambda}) \\
&= [\max_i \delta_t(i)\,a_{ij}]\,b_j(Y_{t+1})
\end{aligned}
$$

References & further reading

References and further reading

Durbin, R., Eddy, S., Krogh, A., Mitchison, G. (1998), *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*, Cambridge University Press.

Ewens, W.J, Grant, G. (2006), *Statistical Methods for Bioinformatics*, Springer, New York.

Fridlyand, J., Snijders, A.M., Pinkel, D., Albertson, D.G., Jain, A.N. (2004), “Hidden Markov models approach to the analysis of CGH data”, *Journal of Multivariate Analysis*, 90(1), 132-153.

Rabiner, L.R. (1989), “A tutorial on hidden Markov models and selected applications in speech recognition”, *Proceedings of the IEEE*, 77(2), 257-286.

Van de Wiel, M.A., Picard, F., Van Wieringen, W.N., Ylstra, B. (2011), "Preprocessing and downstream analysis of microarray DNA copy number profiles", *Briefings in Bioinformatics*, 12(1), 10-21.

Yu, S.-Z. (2010), “Hidden semi-Markov models”, Artificial Intelligence, 174, 215-243.