5 Markov Chains and Hidden Markov Models

We will discuss:

- Markov chains
- Hidden Markov Models (HMMs)
- Profile HMMs

This chapter is based on: S. Durbin, S. Eddy, A. Krogh and G. Mitchison, Biological Sequence Analysis, Cambridge, 1998

5.1 C*p*G-islands

Example: The detection of C*p*G-islands in the human genome.

Double stranded DNA:

```
...ApCpCpApTpGpApTpGpCpApGpGpApCpTpTpCpCpApTpCpGpTpTpCpGpCpGp...
...| | | | | | | | | | | | | | | | | | | | | | | | | | | | |  ...
...TpGpGpTpApCpTpApCpGpTpCpCpTpGpApApGpGpTpApGpCpApApGpCpGpCp...
```

The C in a C*p*G-pair is often modified by methylation (that is, an H-atom is replaced by a CH_3-group). When methylated, there is an increased chance that a C will mutate to a T (spontaneous deamination).

In consequence, as most of the human genome is methylated, C*p*G-pairs are generally under-represented in the human genome.

Upstream of a gene, the methylation process is suppressed in short regions of the genome of length 100-5000, to facilitate transcription. These areas are called C*p*G-*islands* and they are characterized by the fact that there is a higher proportion of C*p*G-pairs then elsewhere.

C*p*G-islands play a role in imprinting and in the deactivation of intra-genomic parasites. Computationally, they are used in gene prediction to separate individual genes.

Definition 5.1.1 (classical definition of C*p*G-islands) *A DNA sequence of length 200 with a* C+G *content of* 50% *and a ratio of observed-to-expected number of* C*p*G*'s that is above* 0.6.[1]

According to one study[2], human chromosomes 21 and 22 contain about 1100 C*p*G-islands and about 750 genes.

We will address the following two main questions concerning C*p*G-islands:

Main questions:

1. Given a short segment of genomic sequence, how to decide whether this segment comes from a C*p*G-island or not?
2. Given a long segment of genomic sequence, how to find all contained C*p*G-islands?

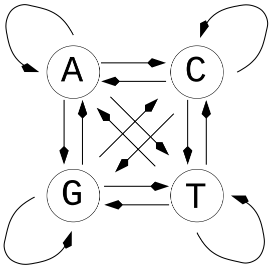
[1] Gardiner-Garden & Frommer, 1987

[2] D. Takai & P. A. Jones, Comprehensive analysis of CpG islands in human chromosomes 21 and 22, PNAS, March 19, 2002

5.2 Markov chains

Our goal is to set up a probabilistic model for `C`p`G`-islands. Because pairs of consecutive nucleotides are important in this context, we need a model in which the probability of one symbol depends on the probability of its predecessor. This leads us to a *Markov chain.*

Example:



Circles= states, e.g. with names `A` , `C` , `G` and `T` .
Arrows= possible transitions, each labeled with a *transition probability* $a_{st} = P(x_i = t \mid x_{i-1} = s)$.

Definition 5.2.1 (Markov chain) *A (time-homogeneous)* Markov chain *(of order 1) is a graphical model* $(\mathcal{S}, A)$ *consisting of a finite set of* states $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ *and a* transition matrix $A = \{a_{st}\}$ *with* $\sum_{t \in \mathcal{S}} a_{st} = 1$ *for all* $s \in \mathcal{S}$*, which determines the probability of the transition* $s \to t$ *as follows:*

$$P(x_{i+1} = t \mid x_i = s) = a_{st}.$$

Time-homogeneous means that probabilities do not change over time.

Order k means that transition probablities depend only states going back k time units; we have $k = 1$.

At any time i the chain is in a specific state x_i and at the tick of a clock the chain probabilistically changes to state x_{i+1} according to the given transition probabilities.

Example: Weather in Tübingen, daily at midday: Possible states are `rain`, `sun` or `clouds`.

Transition probabilities:

today\tomorrow	R	S	C
R	.5	.1	.4
S	.2	.6	.2
C	.3	.3	.4

The probablity that it will be sunny tomorrow, given that it rains today, is 0.1.

Possible sequence of weather states: `...RRRRRRCCSSSSSSCSCSCCCRRCRCSSSS...`

5.2.1 Probability of a sequence of states

Given a sequence of states $x_1, x_2, x_3, \ldots, x_L$, what is the probability that a Markov chain will step through precisely this sequence of states?

$$\begin{aligned} P(x) &= P(x_L, x_{L-1}, \ldots, x_1) \\ &= P(x_L \mid x_{L-1}, \ldots, x_1) P(x_{L-1} \mid x_{L-2}, \ldots, x_1) \ldots P(x_1), \\ & \quad \text{(by repeated application of } P(X,Y) = P(X|Y)P(Y)) \\ &= P(x_L, \mid x_{L-1}) P(x_{L-1} \mid x_{L-2}) \ldots P(x_2 \mid x_1) P(x_1) \\ &= P(x_1) \prod_{i=2}^{L} a_{x_{i-1} x_i}, \end{aligned}$$

because $P(x_i \mid x_{i-1}, \ldots, x_1) = P(x_i \mid x_{i-1}) = a_{x_{i-1} x_i}$, as the Markov chain has order 1.

5.2.2 Modeling the begin and end states

In the previous discussion we overlooked the fact that a Markov chain starts in some state x_1, with initial probability of $P(x_1)$.

We add a *begin state* to the model that is labeled 'b'. We will always assume that $x_0 = \mathtt{b}$ holds. Then:

$$P(x_1 = s) = a_{\mathtt{b}s} = P(s),$$

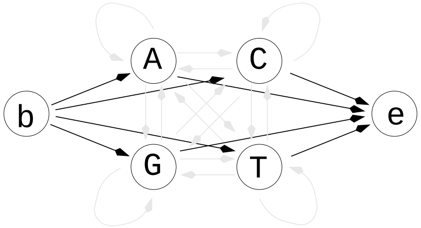
where $P(s)$ denotes the *background probability* of symbol s.

Similarly, we explicitly model the end of the sequence of states using an *end state* 'e'. Thus, the probability that we end in state t (e.g., either A, C, G or T) is

$$P(x_L = t) = a_{x_L \mathtt{e}}.$$

5.2.3 Extension of the model

Example:



```
# Markov chain that generates CpG islands (Source: DEMK98, p 50)
# Number of states:
6
# State labels:
A C G T * +
# Transition matrix:
0.1795 0.2735 0.4255 0.1195 0 0.002
0.1705 0.3665 0.2735 0.1875 0 0.002
0.1605 0.3385 0.3745 0.1245 0 0.002
0.0785 0.3545 0.3835 0.1815 0 0.002
0.2495 0.2495 0.2495 0.2495 0 0.002
0.0000 0.0000 0.0000 0.0000 0 1.000
```

5.2.4 Determining the transition matrix

The transition matrix A^+ is obtained empirically (*trained*) by counting transitions that occur in a training set of known CpG-islands.

This is done as follows:

$$a^+_{st} = \frac{c^+_{st}}{\sum_{t'} c^+_{st'}},$$

where c_{st} is the number of positions in a training set of CpG-islands at which state s is followed by state t.

We obtain A^- empirically in a similar way, using a training set of known non-CpG-islands.

5.2.5 Two examples of Markov chains

```
# Markov chain for CpG islands              # Markov chain for non-CpG islands
# (Source: DEMK98, p 50)                    # (Source: DEMK98, p 50)
# Number of states:                         # Number of states:
6                                           6
```

```
# State labels:
A C G T * +
# Transition matrix:
.1795 .2735 .4255 .1195 0 0.002
.1705 .3665 .2735 .1875 0 0.002
.1605 .3385 .3745 .1245 0 0.002
.0785 .3545 .3835 .1815 0 0.002
.2495 .2495 .2495 .2495 0 0.002
.0000 .0000 .0000 .0000 0 1.000
```

```
# State labels:
A C G T * +
# Transition matrix:
.2995 .2045 .2845 .2095 0 .002
.3215 .2975 .0775 .3015 0 .002
.2475 .2455 .2975 .2075 0 .002
.1765 .2385 .2915 .2915 0 .002
.2495 .2495 .2495 .2495 0 .002
.0000 .0000 .0000 .0000 0 1.00
```

5.2.6 Answering question 1

Suppose we are given a short sequence $x = (x_1, x_2, \dots, x_L)$. Does it come from a C$p$G-island (Model$^+$)?

$$P(x \mid \text{Model}^+) = \prod_{i=0}^{L} a^+_{x_i x_{i+1}},$$

with $x_0 = \texttt{b}$ and $x_{L+1} = \texttt{e}$.

We use the following score:

$$S(x) = \log\left(\frac{P(x \mid \text{Model}^+)}{P(x \mid \text{Model}^-)}\right) = \sum_{i=0}^{L} \log \frac{a^+_{x_i x_{i+1}}}{a^-_{x_i x_{i+1}}}.$$

The higher this score is, the higher the probability is, that x comes from a CpG-island.

5.2.7 Types of questions that a Markov chain can answer

Example weather in Tübingen, daily at midday: Possible states are `rain`, `sun` or `clouds`.

Transition probabilities:

	R	S	C
R	.5	.1	.4
S	.2	.6	.2
C	.3	.3	.4

Types of questions that the model can answer:

If it is sunny today, what is the probability that the sun will shine for the next seven days?

5.3 Hidden Markov Models (HMM)

Main question: Question 2, how to detect CpG-islands inside a long sequence?

One possible approach is a *window technique*: a window of width w is moved along the sequence and the score is plotted.

This approach has problems, such as it is hard to determine the boundaries of CpG-islands, which window size w should one choose?...

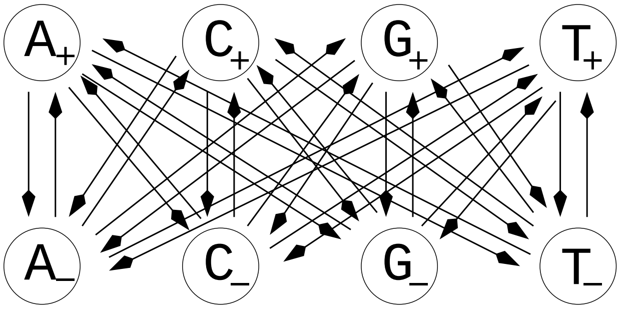
We will consider an alternative approach: Conceptually "merge" the two Markov chains Model$^+$ and Model$^-$ so as to obtain a so-called *Hidden Markov Model*.

Definition 5.3.1 (Hidden Markov Model (HMM)) *A* HMM *is a graphical model* $M = (\mathcal{S}, Q, A, e)$ *consisting of*

- *an alphabet* $\mathcal{S}$,
- *a set of states* Q,
- *a matrix* $A = \{a_{k\ell}\}$ *of transition probabilities* $a_{k\ell}$ *for* $k, \ell \in Q$, *and*
- *an emission probability* $e_k(b)$ *for every* $k \in Q$ *and* $b \in \mathcal{S}$.

5.3.1 Example

The topology of an HMM for `C`p`G`-islands:



(Additionally, we have all transitions between states in either of the two sets that carry over from the two Markov chains Model$^+$ and Model$^-$.)

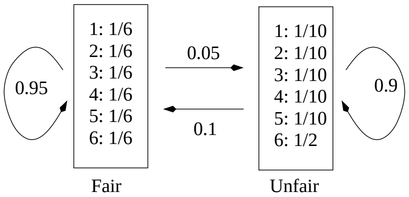
From now on we use 0 for both the begin and end state.

5.3.2 HMM for `C`p`G`-islands

```
# Number of states:
9
# Names of states (begin/end, A+, C+, G+, T+, A-, C-, G- and T-):
0 A C G T a c g t
# Number of symbols:
4
# Names of symbols:
a c g t
# Transition matrix, probability to change from +island to -island (and vice versa) is 10E-4
0.000 0.0725193101 0.1637630296 0.1788242720 0.0754545682 0.1322050994 0.1267006624 0.1226380452 0.1278950131
0.001 0.1762237762 0.2682517483 0.4170629371 0.1174825175 0.0035964036 0.0054745255 0.0085104895 0.0023976024
0.001 0.1672435130 0.3599201597 0.2679840319 0.1838722555 0.0034131737 0.0073453094 0.0054690619 0.0037524950
0.001 0.1576223776 0.3318881119 0.3671328671 0.1223776224 0.0032167832 0.0067732268 0.0074915085 0.0024975025
0.001 0.0773426573 0.3475514486 0.3759440559 0.1781818182 0.0015784216 0.0070929071 0.0076723277 0.0036363636
0.001 0.0002997003 0.0002047952 0.0002837163 0.0002097902 0.2994005994 0.2045904096 0.2844305694 0.2095804196
0.001 0.0003216783 0.0002977023 0.0000769231 0.0003016983 0.3213566434 0.2974045954 0.0778441558 0.3013966034
0.001 0.0002477522 0.0002457542 0.0002977023 0.0002077922 0.2475044955 0.2455084915 0.2974035964 0.2075844156
0.001 0.0001768232 0.0002387612 0.0002917083 0.0002917083 0.1766463536 0.2385224775 0.2914165834 0.2914155844
# Emission probabilities:
0 0 0 0
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

5.3.3 Example fair/loaded dice

Casino uses two dice, *fair* and *loaded*:



Casino guest only observes the number rolled:

`6 4 3 2 3 4 6 5 1 2 3 4 5 6 6 6 3 2 1 2 6 3 4 2 1 6 6...`

Which dice was used remains hidden:

`F F F F F F F F F F F F U U U U U F F F F F F F F F...`

5.3.4 Generation of simulated data

We can use HMMs to generate data:

Algorithm 5.3.2 (Simulator)

```
Start in state 0.
While we have not reentered state 0:
    Choose a new state using the transition probabilities
    Choose a symbol using the emission probabilities and report it.
```

We use the *fair/loaded* HMM to generate a sequence of states and symbols:

```
Symbols: 24335642611341666666526562426612134635535566462666636664253
States : FFFFFFFFFFFFFFUUUUUUUUUUUUUUUUUUFFFFFFFFFFUUUUUUUUUUUUUFFFF

Symbols: 35246363252521655615445653663666511145445656621261532516435
States : FFFFFFFFFFFFFFFFFFFFFFFFFFFUUUUUUUFFUUUUUUUUUUUUUUFFFFFFFFF

Symbols: 5146526666
States : FFUUUUUUUU
```

How probable is a given sequence of data?

If we can observe only the symbols, can we reconstruct the corresponding states?

5.3.5 The probability, given states and symbols

Definition 5.3.3 (Path) *A* path $\pi = (\pi_1, \pi_2, \dots, \pi_L)$ *is a sequence of states in the model* M*.*

Suppose we are given a sequence of symbols $x = (x_1, \dots, x_L)$ and a path $\pi = (\pi_1, \dots, \pi_L)$ through M. The joint probability is:

$$P(x, \pi) = a_{0\pi_1} \prod_{i=1}^{L} e_{\pi_i}(x_i) a_{\pi_i \pi_{i+1}},$$

with $\pi_{L+1} = 0$.

Unfortunately, in practice we usually do not know the path through the model.

5.3.6 "Decoding" a sequence of symbols

Problem: We have observed a sequence x of symbols and would like to "decode" the sequence:

Example: The sequence of symbols `C G C G` has a number of explanations within the `C`p`G`-model, e.g.: (C_+, G_+, C_+, G_+), (C_-, G_-, C_-, G_-) and (C_-, G_+, C_-, G_+).

A path through the HMM determines which parts of the sequence x are classified as `C`p`G`-islands, such a classification of the observed symbols is called a *decoding*.

5.3.7 The most probable path

To solve the decoding problem, we want to determine the path π^* that maximizes the probability of having generated the sequence x of symbols, that is:

$$\pi^* = \arg\max_{\pi} P(x, \pi).$$

This *most probable path* π^* can be computed recursively.

Definition 5.3.4 (Viterbi variable) *Given a prefix $(x_1, x_2, \dots, x_i)$, the* Viterbi *variable $v_k(i)$ denotes the probability that the most probable path is in state k when it generates symbol x_i at position i. Then:*

$$v_\ell(i+1) = e_\ell(x_{i+1}) \max_{k \in Q}(v_k(i) a_{k\ell}),$$

with $v_0(0) = 1$, initially.

(Note: We have: $\arg\max_\pi P(x,\pi) = \arg\max_\pi P(\pi \mid x)$, because $P(x,\pi) = P(\pi \mid x)P(x)$.)

Dynamic programming matrix:

$$\begin{array}{c|cccccc|cc}
x_0 & x_1 & x_2 & x_3 & \dots & x_{i-2} & x_{i-1} & x_i & x_{i+1} \\
 & A_+ & A_+ & A_+ & \dots & \mathbf{A}_+ & A_+ & A_+ & \dots \\
 & C_+ & \mathbf{C}_+ & \mathbf{C}_+ & \dots & C_+ & \mathbf{C}_+ & C_+ & \\
 & \mathbf{G}_+ & G_+ & G_+ & \dots & G_+ & G_+ & \mathbf{G}_+ & \\
 & T_+ & T_+ & T_+ & \dots & T_+ & T_+ & T_+ & \\
\mathbf{0} & A_- & A_- & A_- & \dots & A_- & A_- & A_- & \\
 & C_- & C_- & C_- & \dots & C_- & C_- & C_- & \\
 & G_- & G_- & G_- & \dots & G_- & G_- & G_- & \\
 & T_- & T_- & T_- & \dots & T_- & T_- & T_- &
\end{array}$$

5.3.8 The Viterbi algorithm

Algorithm 5.3.5 (Viterbi algorithm)

Input:	*HMM $M = (\mathcal{S}, Q, A, e)$* *and symbol sequence x*
Output:	*Most probable path π^*.*
Initialization ($i = 0$):	$v_0(0) = 1$, $v_k(0) = 0$ *for* $k \neq 0$.
For all $i = 1 \dots L$, $\ell \in Q$:	$v_\ell(i) = e_\ell(x_i) \max_{k \in Q}(v_k(i-1) a_{k\ell})$ $\mathrm{ptr}_\ell(i) = \arg\max_{k \in Q}(v_k(i-1) a_{k\ell})$
Termination:	$P(x, \pi^*) = \max_{k \in Q}(v_k(L) a_{k0})$ $\pi^*_L = \arg\max_{k \in Q}(v_k(L) a_{k0})$
Traceback: *For all $i = L-1, \dots, 1$:*	$\pi^*_i = \mathrm{ptr}_{\pi^*_{i+1}}(i+1)$

Implementation hint: instead of multiplying many small values, add their logarithms!

5.3.9 Example for Viterbi

Suppose we are given the sequence `C G C G` and the HMM for `C`p`G`-islands. Here is a table of possible values for v:

			sequence			
	v		C	G	C	G
State	0	1	0	0	0	0
	$\mathtt{A}_+$	0	0	0	0	0
	$\mathtt{C}_+$	0	**.16**	0	**.015**	0
	$\mathtt{G}_+$	0	0	**.044**	0	**.0039**
	$\mathtt{T}_+$	0	0	0	0	0
	$\mathtt{A}_-$	0	0	0	0	0
	$\mathtt{C}_-$	0	.13	0	.0026	0
	$\mathtt{G}_-$	0	0	.010	0	.00021
	$\mathtt{T}_-$	0	0	0	0	0

5.3.10 Viterbi-decoding of the casino example

We used the *fair/loaded* HMM to first generate a sequence of symbols and then use the Viterbi algorithm to decode the sequence, result:

```
Symbols: 24335642611341666666526562426612134635535566462666636664253
States : FFFFFFFFFFFFFFUUUUUUUUUUUUUUUUUUFFFFFFFFFFUUUUUUUUUUUUUFFFF
Viterbi: FFFFFFFFFFFFFFUUUUUUUUUUUUUUUUFFFFFFFFFFFFUUUUUUUUUUUUUFFFF

Symbols: 35246363252521655615445653663666511145445656621261532516435
States : FFFFFFFFFFFFFFFFFFFFFFFFFFFUUUUUUUFFUUUUUUUUUUUUUUFFFFFFFFF
Viterbi: FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

Symbols: 5146526666
States : FFUUUUUUUU
Viterbi: FFFFFFUUUU
```

5.3.11 Three Key Questions for HMMs

Let M be a HMM, x a sequence of symbols.

(Q1) For x, determine the most probable sequence of states through M: *Viterbi algorithm*

(Q2) Determine the probability that M generated x: $P(x) = P(x \mid M)$: *forward algorithm*

(Q3) Given x and perhaps some additional sequences of symbols, how do we train the parameters of M? *Baum-Welch* algorithm or *Viterbi training*

5.3.12 Computing $P(x \mid M)$

Suppose we are given an HMM M and a sequence of symbols x. The probability that x was generated by M is given by:

$$P(x \mid M) = \sum_{\pi} P(x, \pi \mid M),$$

summing over all possible state sequences π through M.

5.3.13 Forward algorithm

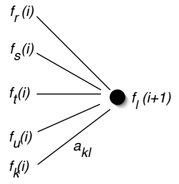
The value of $P(x \mid M)$ can be efficiently computed using the *forward algorithm.*

This algorithm is obtained from the Viterbi algorithm by replacing max by a sum. More precisely, we define the *forward-variable*:

$$f_k(i) = P(x_1 \dots x_i, \pi_i = k),$$

which equals the probability that the model reports the prefix sequence $(x_1, \dots, x_i)$ and is in state $\pi_i = k$ at position i.

We obtain the recursion: $f_\ell(i+1) = e_\ell(x_{i+1}) \sum_{k \in Q} f_k(i) a_{k\ell}$.



Algorithm 5.3.6 (Forward algorithm)

```
Input:                   HMM $M = (\mathcal{S}, Q, A, e)$
                         and sequence of symbols $x$
Output:                  probability $P(x \mid M)$

Initialization ($i = 0$):  $f_0(0) = 1$, $f_k(0) = 0$ for $k \neq 0$.

For all $i = 1 \dots L$, $\ell \in Q$:  $f_\ell(i) = e_\ell(x_i) \sum_{k \in Q}(f_k(i-1) a_{k\ell})$

Result:                  $P(x \mid M) = \sum_{k \in Q}(f_k(L) a_{k0})$
```

Implementation hint: Logarithms can not be employed here easily, but there are so-called "scaling methods".

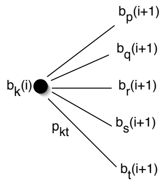
This solves key question Q2!

5.3.14 Backward algorithm

The backward-variable contains the probability to start in state $p_i = k$ and then to generate the suffix sequence $(x_{i+1}, \dots, x_L)$: $b_k(i) = P(x_{i+1} \dots x_L \mid \pi_i = k)$.

Algorithm 5.3.7 (Backward algorithm)

```
Input:                     HMM $M = (\mathcal{S}, Q, A, e)$
                           and sequence of symbols $x$
Output:                    probability $P(x \mid M)$
Initialization ($i = L$):    $b_k(L) = a_{k0}$ for all $k$.
For all $i = L-1 \dots 1$, $k \in Q$:  $b_k(i) = \sum_{\ell \in Q} a_{k\ell} e_\ell(x_{i+1}) b_\ell(i+1)$
Result:                    $P(x \mid M) = \sum_{\ell \in Q}(a_{0l} e_\ell(x_1) b_\ell(1))$
```



5.3.15 Summary of the three variables

Viterbi	$v_k(i)$	probability with which the most probable state path generates the sequence of symbols $(x_1, x_2, \dots, x_i)$ and the system is in state k at time i.
Forward	$f_k(i)$	probability that the prefix sequence of symbols $x_1, \dots, x_i$ is generated, and the system is in state k at time i.
Backward	$b_k(i)$	probability that the system starts in state k at time i and then generates the sequence of symbols $x_{i+1}, \dots, x_L$.

5.3.16 Posterior probabilities

Suppose we are given an HMM M and a sequence of symbols x. Let $P(\pi_i = k \mid x)$ be the probability that symbol x_i was reported in state $\pi_i = k$. We call this the *posterior probability*, as it computed *after* observing the sequence x.

We have:

$$P(\pi_i = k \mid x) = \frac{P(\pi_i = k, x)}{P(x)} = \frac{f_k(i) b_k(i)}{P(x)},$$

as $P(g, h) = P(g \mid h)P(h)$ and by definition of the forward- and backward-variable.

5.3.17 Training the parameters

How does one generate an HMM?

First step: Determine its "topology", i.e. the number of states and how they are connected via transitions of non-zero probability.

The topology is usually designed "by hand".

Second step: Set the parameters, i.e. the transition probabilities $a_{k\ell}$ and the emission probabilities $e_k(b)$.

We will now discuss the second step. Given a set of example sequences, our goal is to train the parameters of the HMM using the example sequences, e.g. to set the parameters so as to maximize the probability with which the HMM generates the given example sequences.

5.3.18 Training when the states are known

Let $M = (\mathcal{S}, Q, A, e)$ be a HMM.

Suppose we are given a list of sequences of symbols $x^1, x^2, \dots, x^n$ and a list of corresponding paths $\pi^1, \pi^2, \dots, \pi^n$. (E.g., DNA sequences with annotated $\mathtt{C}p\mathtt{G}$-islands.)

We want to choose the parameters (A, e) of the HMM M *optimally*, such that:

$$P\big(x^1, \dots, x^n, \pi^1, \dots, \pi^n \mid M = (\mathcal{S}, Q, A, e)\big) =$$

$$\max_{(A', e')} P\big(x^1, \dots, x^n, \pi^1, \dots, \pi^n \mid M = (\mathcal{S}, Q, A', e')\big).$$

In other words, we want to determine the so-called *Maximum Likelihood Estimator* (*ML-estimator*) for (A, e).

5.3.19 ML-Estimation for (A, e)

(Recall: If we consider $P(D \mid M)$ as a function of D, then we call this a *probability*; as a function of M, then we use the word *likelihood*.)

ML-estimation:

$$(A, e)^{\mathrm{ML}} = \arg \max_{(A', e')} P(x^1, \dots, x^n, \pi^1, \dots, \pi^n \mid M = (\mathcal{S}, Q, A', e')).$$

To compute A and e from labeled training data, we first determine the following numbers:

$\hat{a}_{k\ell}$: Number of observed transitions from state k to ℓ
$\hat{e}_k(b)$: Number of observed emissions of b in state k

We then set A and e as follows:

$$a_{k\ell} = \frac{\hat{a}_{k\ell}}{\sum_{q \in Q} \hat{a}_{kq}} \qquad \text{and} \qquad e_k(b) = \frac{\hat{e}_k(b)}{\sum_{s \in \mathcal{S}} \hat{e}_k(s)}. \qquad (*)$$

5.3.20 Training the *fair/loaded* HMM

Suppose we are given example data x and π:

```
Symbols x: 1 2 5 3 4 6 1 2 6 6 3 2 1 5
States pi: F F F F F F F U U U U F F F
```

State transitions:

$\hat{a}_{k\ell}$	0	F	U
0			
F			
U			

→

$a_{k\ell}$	0	F	U
0			
F			
U			

Emissions:

$\hat{e}_k(b)$	1	2	3	4	5	6
0						
F						
U						

→

$e_k(b)$	1	2	3	4	5	6
0						
F						
U						

5.3.21 Pseudocounts

One problem in training is *overfitting*. For example, if some possible transition $k \mapsto \ell$ is never seen in the example data, then we will set $\bar{a}_{k\ell} = 0$ and the transition is then strictly forbidden.

Also, if a given state k is never seen in the example data, then $\bar{a}_{k\ell}$ is undefined for all ℓ.

To solve this problem, we introduce *pseudocounts* $r_{k\ell}$ and $r_k(b)$, and define:

$\hat{a}_{k\ell}$ = number of transitions from k to ℓ in the example data $+ r_{k\ell}$
$\hat{e}_k(b)$ = number of emissions of b in k in the example data $+ r_k(b)$.

Small pseudocounts reflect “little pre-knowledge”, large ones reflect“more pre-knowledge”.

The *Laplace rule* is to use a pseudocount of 1 everywhere.

5.3.22 Viterbi training

In unsupervised training, the usual strategy is to iteratively improve the parameters of the HMM.

One such algorithm is the *Baum Welch* algorithm, which is based on the general technique of *expectation maximization*. It aims at optimizing the log-likelihood score.

Here is a conceptually slightly simpler algorithm, called *Viterbi training*:

Algorithm 5.3.8 (Viterbi training) *Let $M = (\mathcal{S}, Q, A, e)$ be a HMM and assume we are given* training sequences $x^1, x^2, \ldots, x^n$. *For a number of different random seeds, assign random, non-zero transition and emission probabilities to the HMM. Then, iteratively improve the parameters as follows:*

1. *Use the Viterbi algorithm to compute a state path π^i for each of the training sequences x^i.*

2. *Use this data (and pseudo-counts) to modify the parameters of the HMM as in supervised training.*

3. *Repeat until converged.*

Return the parameters (A, e) that produce the highest probabilities on the training data.

5.4 Protein Families

Suppose we are given the following related sequences, how to characterize this "family"?

```
#A-helices  ...........AAAAAAAAAAAAAAAA...BBBBBBBBBBBBBBBBCCCCCCCCCCC....DDDDDDDEEEEEEEEEEEE
GLB1_GLYDI  .........GLSAAQRQVIAATWKDIAGADNGAGVGKDCLIKFLSAHPQMAAVFG.FSG....AS...DPGVAALGAKVL
HBB_HUMAN   ........VHLTPEEKSAVTALWGKV....NVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKVKAHGKKVL
HBA_HUMAN   .........VLSPADKTNVKAAWGKVGA..HAGEYGAEALERMFLSFPTTKTYFPHF.DLS.....HGSAQVKGHGKKVA
MYG_PHYCA   .........VLSEGEWQLVLHVWAKVEA..DVAGHGQDILIRLFKSHPETLEKFDRFKHLKTEAEMKASEDLKKHGVTVL
GLB5_PETMA  PIVDTGSVAPLSAAEKTKIRSAWAPVYS..TYETSGVDILVKFFTSTPAAQEFFPKFKGLTTADQLKKSADVRWHAERII
GLB3_CHITP  ..........LSADQISTVQASFDKVKG......DPVGILYAVFKADPSIMAKFTQFAG.KDLESIKGTAPFETHANRIV
LGB2_LUPLU  ........GALTESQAALVKSSWEEFNA..NIPKHTHRFFILVLEIAPAAKDLFS.FLK.GTSEVPQNNPELQAHAGKVF

#A-helices  EEEEEEEEE............FFFFFFFFFFFF..FFGGGGGGGGGGGGGGGGGGG.....HHHHHHHHHHHHHHHHHHH
GLB1_GLYDI  AQIGVAVSHL..GDEGKMVAQMKAVGVRHKGYGNKHIKAQYFEPLGASLLSAMEHRIGGKMNAAAKDAWAAAYADISGAL
HBB_HUMAN   GAFSDGLAHL...D..NLKGTFATLSELHCDKL..HVDPENFRLLGNVLVCVLAHHFGKEFTPPVQAAYQKVVAGVANAL
HBA_HUMAN   DALTNAVAHV...D..DMPNALSALSDLHAHKL..RVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFLASVSTVL
MYG_PHYCA   TALGAILKK....K.GHHEAELKPLAQSHATKH..KIPIKYLEFISEAIIHVLHSRHPGDFGADAQGAMNKALELFRKDI
GLB5_PETMA  NAVNDAVASM..DDTEKMSMKLRDLSGKHAKSF..QVDPQYFKVLAAVIADTVAG........DAGFEKLMSMICILL
GLB3_CHITP  GFFSKIIGEL..P...NIEADVNTFVASHKPRG...VTHDQLNNFRAGFVSYMKAHT..DFA.GAEAAWGATLDTFFGMI
LGB2_LUPLU  KLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG...VADAHFPVVKEAILKTIKEVVGAKWSEELNSAWTIAYDELAIVI

#A-helices  HHHHHHH....
GLB1_GLYDI  ISGLQS.....
HBB_HUMAN   AHKYH......
HBA_HUMAN   TSKYR......
MYG_PHYCA   AAKYKELGYQG           Alignment of seven globin sequences
GLB5_PETMA  RSAY.......           How can this family be characterized?
GLB3_CHITP  FSKM.......
LGB2_LUPLU  KKEMNDAA...
```

Some ideas for characterizing a family:

- Exemplary sequence
- Consensus sequence
- Multiple sequence alignment
- Regular expression (Prosite):

```
LGB2_LUPLU   ...FNA--NIPKH...
GLB1_GLYDI   ...IAGADNGAGV...

...[FI]-[AN]-[AG]-x(1,2)-N-[IG]-[AP]-[GK]-[HV]...
```

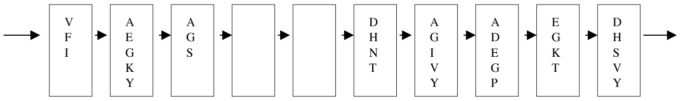
- HMM?

5.4.1 Simple HMM

How to represent the following family of sequences?

```
HBA_HUMAN    ...VGA--HAGEY...
HBB_HUMAN    ...V----NVDEV...
MYG_PHYCA    ...VEA--DVAGH...
GLB3_CHITP   ...VKG------D...
GLB5_PETMA   ...VYS--TYETS...
LGB2_LUPLU   ...FNA--NIPKH...
GLB1_GLYDI   ...IAGADNGAGV...
```

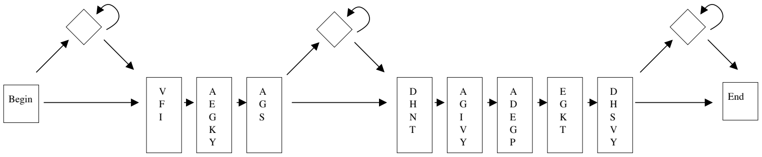
We first consider a simple HMM that is equivalent to a PSSM (*Position Specific Score Matrix*):



(The listed amino-acids have a higher emission-probability.)

5.4.2 Insert-states

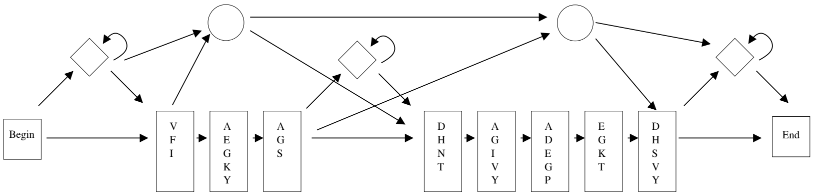
We introduce so-called *insert*-states that emit symbols based on their background probabilities.



This allows us to model segments of sequence that lie outside of conserved domains.

5.4.3 Delete-states

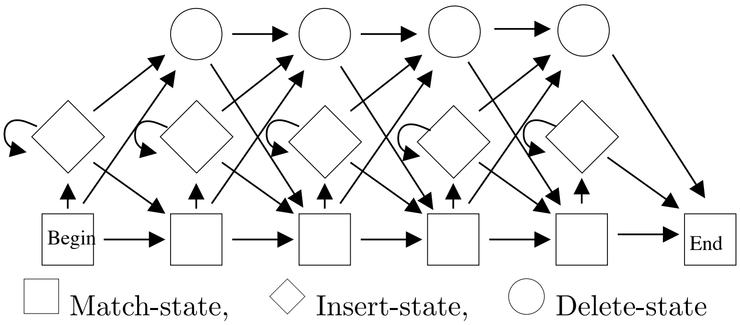
We introduce so-called *delete*-states that are silent and do not emit any symbols.



This allows us to model the absence of individual domains.

5.4.4 Topology of a profile-HMM

The result is a so-called *profile HMM*:



5.4.5 Design of a profile-HMM

Suppose we are given a multiple alignment of a family of sequences.

First we must decide which positions are to be modeled as match- and which positions are to be modeled as insert-states. Rule-of-thumb: columns with more than 50% gaps should be modeled as insert-states.

We determine the transition and emission probabilities simply by counting the observed transitions $A_{k\ell}$ and emissions $E_k(B)$:

$$a_{k\ell} = \frac{A_{k\ell}}{\sum_{\ell'} A_{k\ell'}} \text{ and } e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}.$$

It may happen that certain transitions or emissions do not appear in the training data and so we use the *Laplace rule* and add 1 to each count.

5.5 Summary

A Hidden Markov model (HMM) is an example of a machine-learning datastructure that can be used to classify sequences. It is applicable when:

- there are a sequential dependencies in the data, and
- there is a sufficient supply of training sequences.

Training is known as *supervised*, if annotated training data are available, or *unsupervised*, if the training data are not annotated.

HMMs are used in the detection of `C`p`G`-islands and, more importantly, as profile HMMs to define protein families. Another application is in gene prediction.