9 Markov chains and Hidden Markov Models

We will discuss:

- Markov chains
- Hidden Markov Models (HMMs)
- Algorithms: Viterbi, forward, backward, posterior decoding
- Profile HMMs
- Baum-Welch algorithm

This chapter is based on:

- R. Durbin, S. Eddy, A. Krogh, G. Mitchison: Biological sequence analysis. Cambridge University Press, 1998. ISBN 0-521-62971-3 (Chapter 3)
- An earlier version of this lecture by Daniel Huson.
- Lecture notes by Mario Stanke, 2006.

9.1 CpG-islands

As an introduction to Markov chains, we consider the problem of finding CpG-islands in the human genome.

A piece of double stranded DNA:

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

The C in a CpG pair is often modified by *methylation* (that is, an H-atom is replaced by a CH_3-group). There is a relatively high chance that the methyl-C will mutate to a T. Hence, CpG-pairs are underrepresented in the human genome.

Methylation plays an important role in transscription regulation. Upstream of a gene, the methylation process is suppressed in a short region of length 100-5000. These areas are called CpG-*islands*. They are characterized by the fact that we see more CpG-pairs in them than elsewhere.

Therefore CpG-islands are useful markers for genes in organisms whose genomes contain 5-methyl-cytosine.

CpG-islands in the promoter-regions of genes play an important role in the deactivation of one copy of the X-chromosome in females, in genetic imprinting and in the deactivation of intra-genomic parasites.

Classical definition: DNA sequence of length 200 with a C + G content of 50% and a ratio of observed-to-expected number of CpG's that is above 0.6. (Gardiner-Garden & Frommer, 1987)

According to a recent study, human chromosomes 21 and 22 contain about 1100 CpG-islands and about 750 genes. (Comprehensive analysis of CpG islands in human chromosomes 21 and 22, D. Takai & P. A. Jones, PNAS, March 19, 2002)

More specifically, we can ask the following

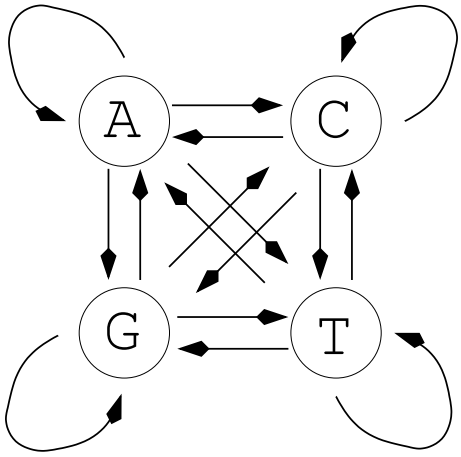
Questions.

1. Given a (short) segment of genomic sequence.
   How to *decide* whether this segment is from a `CpG`-island or not?
2. Given a (long) segment of genomic sequence.
   How to *find* all `CpG`-islands contained in it?

9.2 Markov chains

Our goal is to come up with a probabilistic model for `CpG`-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 dependency is captured by the concept of 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}. Let x_i denote the state at time i. Then $a_{st} := P(x_{i+1} = t \mid x_i = s)$ is the conditional probability to go to state t in the next step, given that the current state is s.

Definition.
A (time-homogeneous) *Markov chain* (of order 1) is a system (Q, A) consisting of a finite set of *states* $Q = \{s_1, s_2, \ldots, s_n\}$ and a *transition matrix* $A = \{a_{st}\}$ with $\sum_{t \in Q} a_{st} = 1$ for all $s \in Q$ that determines the probability of the transition $s \to t$ by

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

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

Remarks on terminology.

- *Order 1* means that the transition probabilities of the Markov chain can only "remember" *1* state of its history. Beyond this, it is *memoryless*. The "memorylessness" condition is a very important. It is called the *Markov property*.
- The Markov chain is *time-homogenous* because the transition probability

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

does not depend on the time parameter i.

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	.5	.3
C	.3	.3	.4

Note that all *rows* add up to 1.

Weather: `...rrrrrrccsssssscscscccrrcrcssss...`

Given a sequence of states $s_1, s_2, s_3, \ldots, s_L$. What is the probability that a Markov chain $x = x_1, x_2, x_3, \ldots, x_L$ will step through precisely this sequence of states? We have

$$\begin{aligned} &P(x_L = s_L, x_{L-1} = s_{L-1}, \ldots, x_1 = s_1) \\ &\quad = P(x_L = s_L \mid x_{L-1} = s_{L-1}, \ldots, x_1 = s_1) \\ &\qquad \cdot P(x_{L-1} = s_{L-1} \mid x_{L-2} = s_{L-2}, \ldots, x_1 = s_1) \\ &\qquad \vdots \\ &\qquad \cdot P(x_2 = s_2 \mid x_1 = s_1) \\ &\qquad \cdot P(x_1 = s_1) \end{aligned}$$

using the "expansion"

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} \iff P(A \cap B) = P(A \mid B) \cdot P(B)\,.$$

Now, we make use of the fact that

$$P(x_i = s_i \mid x_{i-1} = s_{i-1}, \ldots, x_1 = s_1) = P(x_i = s_i \mid x_{i-1} = s_{i-1})$$

by the Markov property. Thus

$$\begin{aligned} &P(x_L = s_L, x_{L-1} = s_{L-1}, \ldots, x_1 = s_1) \\ &\quad = P(x_L = s_L \mid x_{L-1} = s_L - 1) \\ &\qquad \cdot P(x_{L-1} = s_{L-1} \mid x_{L-2} = s_{L-2}) \\ &\qquad \cdot \ldots \cdot P(x_2 = s_2 \mid x_1 = s_1) \cdot P(x_1 = s_1) \\ &\quad = P(x_1 = s_1) \prod_{i=2}^{L} a_{s_{i-1} s_i}\,. \end{aligned}$$

Hence:

The probability of a path is the product of the probability of the initial state and the transition probabilities of its edges.

9.3 Modeling the begin and end states

A Markov chain *starts* in state x_1 with an initial probability of $P(x_1 = s)$. For simplicity (i.e., uniformity of the model) we would like to model this probability as a transition, too.

Therefore we add a *begin state* to the model that is labeled 'b'. We also impose the constraint that $x_0 = \mathtt{b}$ holds. Then:

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

This way, we can store all probabilities in one matrix and the "first" state x_1 is no longer special:

$$P(x_L = s_L, x_{L-1} = s_{L-1}, \ldots, x_1 = s_1) = \prod_{i=1}^{L} a_{s_{i-1}s_i}\,.$$

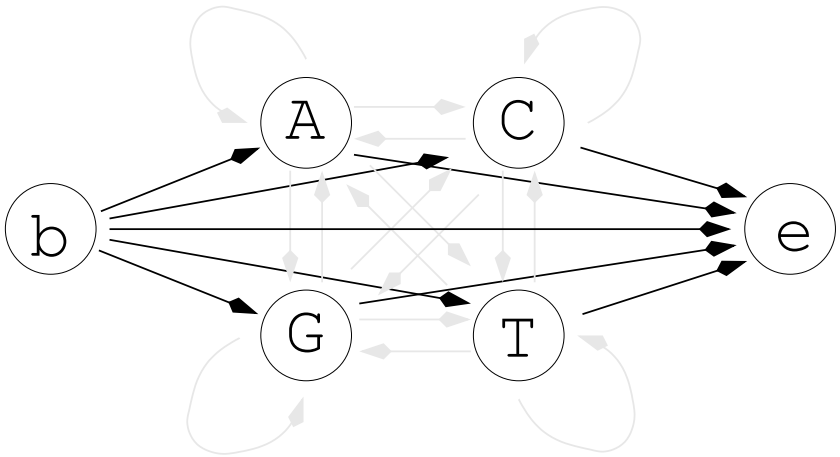
Similarly, we explicitly model the *end* of the sequence of states using an *end state* 'e'. Thus, the probability that the Markov chain stops is

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

if the current state is t.

We think of b and e as *silent* states, because they do not correspond to letters in the sequence. (More applications of silent states will follow.)

Example:




```
# Markov chain that generates CpG islands
# (Source: DEMK98, p 50)
# Number of states:
6
# State labels:
A,  C,  G,  T,  *=b,  +=e
# 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
```

A word on finite automata and regular grammars: One can view Markov chains as nondeterministic finite automata where each transition is also assigned a probability. The analogy also translates to grammars: A stochastic regular grammar is a regular grammar where each production is assigned a probability.

9.4 Determining the transition matrix

How do we find transition probabilities that explain a given set of sequences best?

The transition matrix A^+ for DNA that comes from a CpG-island, is determined 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 can calculate these counts in a single pass over the sequences and store them in a $\Sigma \times \Sigma$ matrix.

We obtain the matrix A^- for non-CpG-islands from empirical data in a similar way.

In general, the matrix of transition probabilities is not symmetric.

Two examples of Markov chains.

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

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

Note the different values for CpG: $a^+_{\mathtt{CG}} = 0.2735$ versus $a^-_{\mathtt{CG}} = 0.0775$.

9.5 Testing hypotheses

When we have two models, we can ask which one explains the observation better.

Given a (short) sequence $x = (x_1, x_2, \ldots, x_L)$. Does it come from a CpG-island (model$^+$)?

We have

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

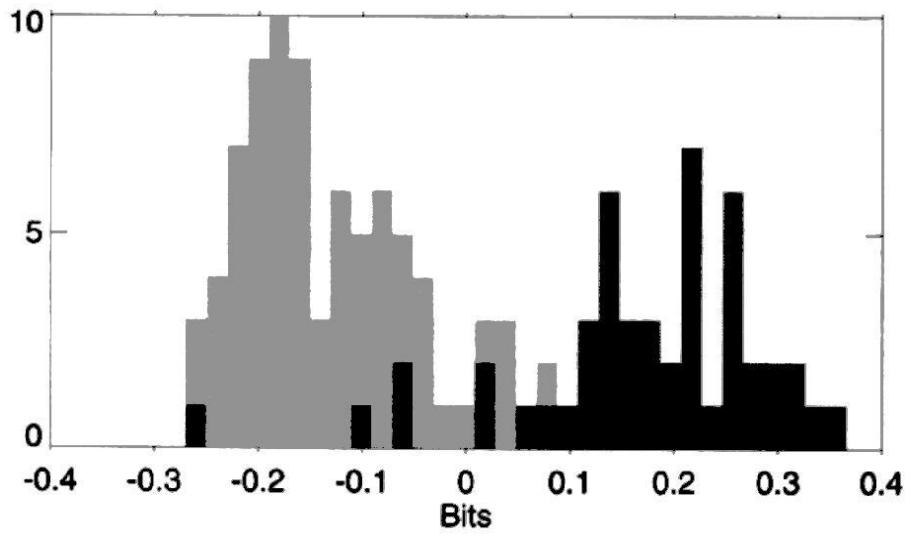
with $x_0 = \mathtt{b}$ and $x_{L+1} = \mathtt{e}$. Similar for (model$^-$).

To compare the models, we calculate the *log-odds ratio*:

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

Then this ratio is normalized by the length of x. This resulting *length-normalized log-odds score* $S(x)/|x|$ can be used to classify x. The higher this score is, the higher the probability is that x comes from a CpG-island.

The histogram of the length-normalized scores for the sequences from the training sets for A^+ and A^- shows that $S(x)/|x|$ is indeed a good classifier for this data set. (Since the base two logarithm was used, the unit of measurement is called "bits".)



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

Types of questions that the Markov chain model can answer:

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

And what is more unlikely: 7 days sun or 8 days rain?

9.6 Hidden Markov Models (HMMs)

Motivation: Question 2, how to *find* `CpG`-islands in a long sequence?

We could approach this using Markov Chains and a "window technique": a window of width w is moved along the sequence and the score (as defined above) is plotted. However the results are somewhat unsatifactory: It is hard to determine the boundaries of CpG-islands, and which window size w should one choose? ...

The basic idea is to relax the tight connection between "states" and "symbols". Instead, every state can "emit" every symbol. There is an emission probability $e_k(b)$ for each state k and symbol b.

However, when we do so, we no longer know for sure the state from which an observed symbol was emitted. In this sense, the Markov model is hidden; hence the name.

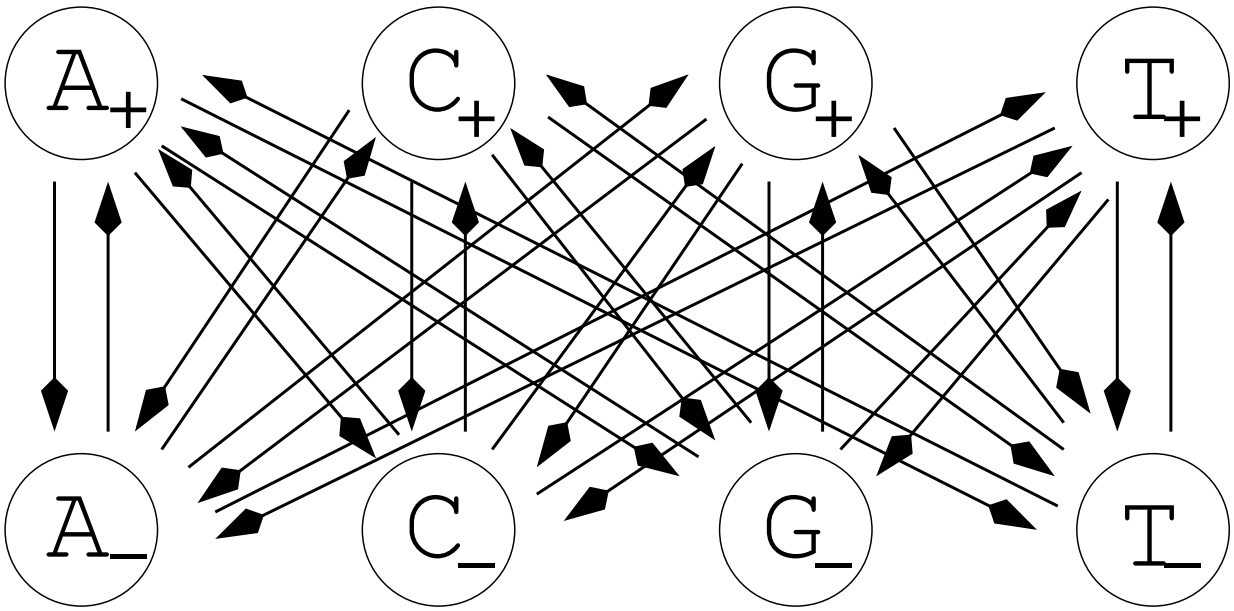
Definition.
Am *HMM* is a system $M = (\Sigma, Q, A, e)$ consisting of

- an *alphabet* Σ,
- a set of *states* Q,

- a matrix $A = \{a_{kl}\}$ of *transition probabilities* a_{kl} for $k, l \in Q$, and
- an *emission probability* $e_k(b)$ for every $k \in Q$ and $b \in \Sigma$.

9.7 HMM for CpG-islands

For example, we can "merge" the two Markov chains model$^+$ and model$^-$ to obtain an HMM for CpG-islands:



We have added all transitions between states in either of the two sets that carry over from the two Markov chains model$^+$ and model$^-$. *The old edges within the two models are still there, but not shown here.*

```
# 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:
#     0         A+        C+        G+        T+         A-        C-        G-        T-
0   0.0000000 0.0725193 0.1637630 0.1788242 0.0754545  0.1322050 0.1267006 0.1226380 0.1278950
A+  0.0010000 0.1762237 0.2682517 0.4170629 0.1174825  0.0035964 0.0054745 0.0085104 0.0023976
C+  0.0010000 0.1672435 0.3599201 0.2679840 0.1838722  0.0034131 0.0073453 0.0054690 0.0037524
G+  0.0010000 0.1576223 0.3318881 0.3671328 0.1223776  0.0032167 0.0067732 0.0074915 0.0024975
T+  0.0010000 0.0773426 0.3475514 0.3759440 0.1781818  0.0015784 0.0070929 0.0076723 0.0036363
A-  0.0010000 0.0002997 0.0002047 0.0002837 0.0002097  0.2994005 0.2045904 0.2844305 0.2095804
C-  0.0010000 0.0003216 0.0002977 0.0000769 0.0003016  0.3213566 0.2974045 0.0778441 0.3013966
G-  0.0010000 0.0001768 0.0002387 0.0002917 0.0002917  0.1766463 0.2385224 0.2914165 0.2914155
T-  0.0010000 0.0002477 0.0002457 0.0002977 0.0002077  0.2475044 0.2455084 0.2974035 0.2075844
# Emission probabilities:
#   a c g t
0   0 0 0 0
A+  1 0 0 0
C+  0 1 0 0
G+  0 0 1 0
T+  0 0 0 1
A-  1 0 0 0
C-  0 1 0 0
G-  0 0 1 0
T-  0 0 0 1
```

Remark: The transition probabilities are usually written in matrix form. It is convenient to have the same index set for rows and columns. Sometimes the symbol 0 is used for the begin *and* end state (as seen above). The meaning should be clear from the context.

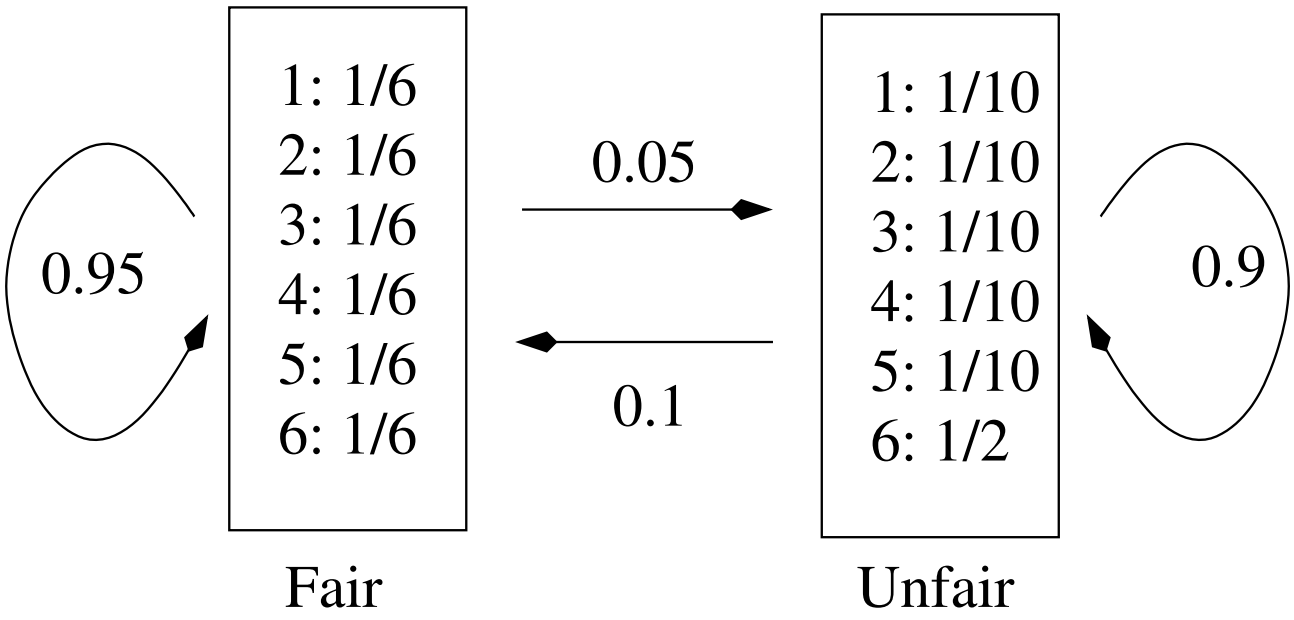
Note the emission probabilities: The model emits the letters `A`, `C`, `G`, `T`, but for each letter there are two states where the letter can come from. Thus we cannot reconstruct the path the HMM has taken from the sequence alone.

In general, the emission probabilities need not be zero-one. But in the HMM for `CpG`-islands every state emits a unique letter. Thus the emitted symbol is not really "random" for a given state.

Next we look at an example where the states and the emitted symbols are associated in a looser way.

9.8 Example: fair/loaded die

An *occasionally dishonest casino* uses two dice, a *fair* and a *loaded* one:



A casino guest only observes the *numbers* 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 ...`

However, which *die* was used remains hidden:
`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 F ...`

9.9 Generation of simulated data

HMMs, like Markov chains, are related to stochastic regular grammars and finite automata. Here is how we can generate a random sequence using an HMM:

Algorithm: Random sequence from HMM

- Start in state 0.
- While we have not re-entered state 0:
  - Choose a new *state* according to the transition probabilities.
  - Choose a *symbol* using the emission probabilities, and report it.

Example.
Here the fair/loaded HMM was used to generate a sequence of states and symbols:

```
States : FFFFFFFFFFFFFFUUUUUUUUUUUUUUUUUUUUUFFFFFFFFFFFUUUUUUUUUUUUUUUFFFF
Symbols: 24335642611341666666526562426612134635535566462666636664253

States : FFFFFFFFFFFFFFFFFFFFFFFFFFFUUUUUUUFFUUUUUUUUUUUUUUUFFFFFFFFF
Symbols: 35246363252521655615445653663666511145445656621261532516435

States : FFUUUUUUUU
Symbols: 5146526666
```

Questions.

- Given an HMM and a sequence of states and symbols. What is the *probability* to get this sequence?
- Given an HMM and a sequence of symbols. Can we *reconstruct* the corresponding sequence of states, assuming that the sequence was generated using the HMM?

9.10 Probability for given states and symbols

Definitions.

- A *path* $\pi = (\pi_1, \pi_2, \ldots, \pi_L)$ in an HMM $M = (\Sigma, Q, A, e)$ is a sequence of states $\pi_i \in Q$.
- Given a sequence of *symbols* $x = (x_1, \ldots, x_L)$ and a *path* $\pi = (\pi_1, \ldots, \pi_L)$ through M. Then 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 $\pi_{L+1} = 0$.

Schematically,

$$\text{begin} \to \underset{a_{0,\pi_1}}{\text{transition}} \to \underset{e_{\pi_1}(x_1)}{\text{emission}} \to \underset{a_{\pi_1,\pi_2}}{\text{transition}} \to \underset{e_{\pi_2}(x_2)}{\text{emission}} \to \cdots \to \underset{a_{\pi_L,0}}{\text{transition}} \to \text{end}$$

All we need to do is multiply these probabilities .

This answers the first question. However, usually we do *not know* the path π through the model! That information is *hidden*.

9.11 The decoding problem

The *decoding problem* is the following: We have observed a sequence x of symbols that was generated by an HMM and we would like to "decode" the sequence of states from it.

Example: The sequence of symbols `CGCG` has a large number of possible "explanations" within the `CpG`-model, including e.g.:

(C_+, G_+, C_+, G_+), (C_-, G_-, C_-, G_-) and (C_-, G_+, C_-, G_+).

Among these, the first one is more likely than the second. The third one is very unlikely because the "signs" alternate, and those transitions have small probabilities.

A path through the HMM determines which parts of the sequence x are classified as `CpG`-islands $(+/-)$. Such a classification of the observed symbols is also called a decoding. But here we will only consider the case where we want to reconstruct the passed states themselves, and not a "projection" of them.

Another example.
In *speech recognition*, HMMs have been applied since the 1970s. One application is the following. A speech signal is sliced into pieces of 10-20 milliseconds, each of which is assigned to one of e.g. 256 categories. We want to find out what sequence of phonemes was spoken. Since the pronounciation of phonemes in natural language varies a lot, we are faced with a decoding problem.

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:

$$\pi^* = \arg\max_\pi \Pr(\pi \mid x) = \arg\max_\pi \Pr(\pi, x).$$

For a sequence of n symbols there are $|Q|^n$ possible paths, therefore we cannot solve the problem by full enumeration.

Luckily, the "most probable path" π^* can be computed by *dynamic programming*. The recursion involves the following entities:

Definition. Given a prefix $(x_1, x_2, \ldots, x_i)$ of the sequence x which is to be decoded. Then let $(\pi_1^*, \pi_2^*, \ldots, \pi_i^*)$ be a *path of states* with $\pi_i^* = s$ which maximizes the probability that the HMM followed theses states and emitted the symbols $(x_1, x_2, \ldots, x_i)$ along its way. That is,

$$(\pi_1^*, \ldots, \pi_i^*) = \arg\max \Big\{ \prod_{k=1}^{i} a_{\pi_{i-1},\pi_i} e_{\pi_i}(x_i) \;\Big|\; (\pi_1, \ldots, \pi_i) \in Q^i, \pi_i = s, \pi_0 = 0 \Big\}.$$

Also, let $V(s, i)$ denote the *value* of this maximal probability. These are sometimes called *Viterbi variables*.

Clearly we can store the values $V(s, i)$ in a $Q \times [1 \,..\, L]$ dynamic programming matrix.

Initialization.
Every path starts at state 0 with probability 1. Hence, the *initialization* for $i = 0$ is $V(0, 0) = 1$, and $V(s, 0) = 0$ for $s \in Q \setminus \{0\}$.

Recursion.
Now for the *recurrence formula*, which applies for $i = 1, \ldots, L$. Assume that we know the most likely path for $x_1, \ldots, x_{i-1}$ under the additional constraint that the last state is s, for all $s \in Q$. Then we obtain the most likely path to the i-th state t by maximizing the probability $V(s, i-1)a_{st}$ over all predecessors $s \in Q$ of t. To obtain $V(t, i)$ we also have to multiply by $e_t(x_i)$ since we have to emit the given symbol x_i.

That is, we have

$$V(t, i) = \max\{\, V(s, i-1)a_{st} \mid s \in Q \,\} \cdot e_t(x_i)$$

for all $t \in Q$. (Again, note the use of the Markov property!)

Termination.
In the last step, we enter state 0 but do not emit a symbol. Hence $P(x, \pi^*) = \max\{\, V(s, L)a_{s,0} \mid s \in Q \,\}$.

9.12 Viterbi algorithm

Input: HMM $M = (\Sigma, Q, A, e)$
and symbol sequence x

Output: Most probable path π^*.

Initialization ($i = 0$)*:* $V(0,0) = 1$, $V(s,0) = 0$ for $s \in Q \setminus \{0\}$.

Recurrence:

For $i = 1, \ldots, L$, $t \in Q$: $V(t,i) = e_t(x_i) \max\{ V(s,i-1)a_{s,t} \mid s \in Q \}$
$T(t,i) = \arg\max\{ V(s,i-1)a_{s,t} \mid s \in Q \}$

Termination ($i = L+1$)*:* $P(x,\pi^*) = \max\{ V(s,L)a_{s,0} \mid s \in Q \}$
$\pi^*_L = \arg\max\{ V(s,L)a_{s,0} \mid s \in Q \}$

Traceback:

For $i = L+1, \ldots, 1$: $\pi^*_{i-1} = T(\pi^*_i, i)$

x_0	x_1	x_2	x_3	...	x_{i-2}	x_{i-1}	x_i	...	x_L	x_{L+1}
0				...				...		**0**
	A_+	A_+	A_+	...	$\mathbf{A}_+$	A_+	A_+	...	A_+	
	C_+	$\mathbf{C}_+$	$\mathbf{C}_+$	...	C_+	$\mathbf{C}_+$	C_+	...	C_+	
	$\mathbf{G}_+$	G_+	G_+	...	G_+	G_+	$\mathbf{G}_+$	...	G_+	
	T_+	T_+	T_+	...	T_+	T_+	T_+	...	$\mathbf{T}_+$	
	A_-	A_-	A_-	...	A_-	A_-	A_-	...	A_-	
	C_-	C_-	C_-	...	C_-	C_-	C_-	...	C_-	
	G_-	G_-	G_-	...	G_-	G_-	G_-	...	G_-	
	T_-	T_-	T_-	...	T_-	T_-	T_-	...	T_-	

We need not consider the values in the 0 row except for the initialization and the termination of the DP.

The *running time* is $|Q|^2|L|$, as each entry of the $Q \times L$ matrix requires $|Q|$ calculations.

The repeated multiplication of probabilities will quickly produce very small numbers. In order to avoid *underflow arithmetic errors*, the calculations in the Viterbi algorithm should therefore be done in "log scale", i.e., we store $\log V(s,i)$ instead of $V(s,i)$. This makes multiplications become additions (which are calculated faster), and the numbers stay in a reasonable range. Moreover, $\log a_{s,t}$ and $\log e_t(y)$ might be rounded to discrete steps so that we can use (unsigned) integer arithmetic.

E.g. a probability $p = 10^{-12345}$ might be stored as $-\lfloor 1000 \cdot \log_{10} p \rfloor = 12345000$.

9.13 Examples for Viterbi

Given the sequence CGCG and the HMM for CpG-islands. Here is the DP table V. (Some traceback pointers are indicated in [ ].):

state		sequence				
		C	G	C	G	
0	1					[G_+] **0.0000039**
A_+	0	0	0	0	0	
C_+	0	[0] **0.16**	0	[G_+] **0.015**	0	
G_+	0	0	[C_+] **0.044**	0	[C_+] **0.004**	
T_+	0	0	0	0	0	
A_-	0	0	0	0	0	
C_-	0	[0] 0.13	0	[G_-] 0.0025	0	
G_-	0	0	[C_-] 0.010	0	[C_-] 0.00019	
T_-	0	0	0	0	0	

So this appears to be from a CpG_+-island ...

The fair/loaded HMM was used to *generate* a sequence of symbols and then the Viterbi-algorithm to *decode* the sequence. The result is:

```
True state:  FFFFFFFFFFFFFFUUUUUUUUUUUUUUUUUUUUFFFFFFFFFFUUUUUUUUUUUUUUFFFF
Symbols:     24335642611341666666526562426612134635535566462666636664253
Viterbi:     FFFFFFFFFFFFFFUUUUUUUUUUUUUUUUUUUUFFFFFFFFFFUUUUUUUUUUUUUUFFFF

True state:  FFFFFFFFFFFFFFFFFFFFFFFFFFFFUUUUUUUUFFUUUUUUUUUUUUUUUFFFFFFFFF
Symbols:     35246363252521655615445653663666511145445656621261532516435
Viterbi:     FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

True state:  FFUUUUUUUUU
Symbols:     5146526666
Viterbi:     FFFFFFUUUU
```

9.14 Algorithmic tasks for HMMs

Let M be an HMM and x be a sequence of symbols.

1. Determine the sequence of states through M which has the highest probability to generate x: *Viterbi algorithm*
2. Determine the probability $P(x)$ that M generates x: *forward algorithm* or *backward algorithm*
3. Assuming that x was generated by M, determine the conditional probability that M was in state s when it generated the i-th symbol x_i: *Posterior decoding*, done by a combination of the *forward and backward algorithm*
4. Given x and perhaps some additional sequences of symbols, adjust the transition and emission probabilities of M such that it explains best these observations: E.g., *Baum-Welch algorithm*

9.15 Forward algorithm

We have already seen a closed formula for the probability $P(x, \pi)$ that M generated x using the path π. Summing over all possible paths, we obtain the probability that M generated x:

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

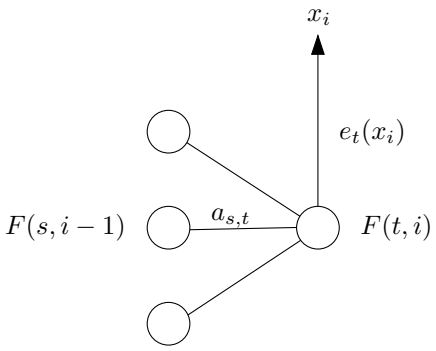
Calculating this sum is done by "replacing the max with a sum" in the Viterbi algorithm.

Formally, we define the *forward-variable*:

$$F(s, i) = \Pr(x_1 \dots x_i, \pi_i = s),$$

This is the probability that the sequence generated by the HMM has the sequence $(x_1, \dots, x_i)$ as a prefix, and the i-th state is $\pi_i = s$. Using the distributive law (and a similar reasoning as for $\Pr(x, \pi)$), we obtain the recursion

$$F(t, i) = e_t(x_i) \cdot \sum_{s \in Q} F(s, i-1) a_{s,t} .$$

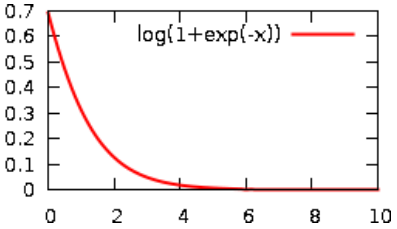


The resulting algorithm is actually simpler, since we do not have to maintain backtrace pointers:

Input:	HMM $M = (\Sigma, Q, A, e)$
	and symbol sequence x
Output:	Probability $\Pr(x)$.
Initialization ($i = 0$)*:*	$F(0, 0) = 1$, $F(s, 0) = 0$ for $s \in Q \setminus \{0\}$.
Recurrence:	
For all $i = 1 \dots L$, $t \in Q$:	$F(t, i) = e_t(x_i) \sum_{s \in Q} F(s, i-1) a_{st}$
Termination ($i = L + 1$)*:*	$\Pr(x) = \sum_{s \in Q} F(s, L) a_{s,0}$

The "log-scale" transform is a bit more complicated in this case, since we need to *add* probabilities in some places. But there are efficient ways to calculate $\log(p + q)$ from $\log p$ and $\log q$: Let $r := p + q$, w.l.o.g. $p > q$. Then

$$\begin{aligned} \log r &= \log(e^{\log p} + e^{\log q}) \\ &= \log\left(e^{\log p} \cdot (1 + e^{\log q} / e^{\log p})\right) \\ &= \log(e^{\log p}) + \log(1 + e^{\log q} / e^{\log p}) \\ &= \log p + \log(1 + e^{\log q - \log p}) \end{aligned}$$



and in this way it can be calculated using a numerical approximation for the function $x \mapsto \log(1 + e^{-x})$, $x \geq 0$. (Take $x := \log p - \log q$.)

Alternatively, one can apply *scaling methods*. The idea is to multiply all entries of a column of the DP matrix with the same factor, if they become too small.

9.16 Backward algorithm

Recall the definition of the *forward-variables*:

$$F(s,i) = \Pr(x_1 \dots x_i, \pi_i = s)\,.$$

This is the probability that the sequence generated by the HMM has the sequence $(x_1, \dots, x_i)$ as a *prefix*, and the i-th state is $\pi_i = s$.

For the *posterior decoding* (described later) we need to compute the conditional probability that M emitted x_i from state π_i, when it happened to generate x:

$$\Pr(\pi_i = s \mid x) = \frac{\Pr(\pi_i = s, x)}{\Pr(x)}\,.$$

Since $\Pr(x)$ is known by the forward algorithm, it suffices to calculate $\Pr(\pi_i = s, x)$. We have

$$\begin{aligned}\Pr(x, \pi_i = s) &= \Pr(x_1, \dots, x_i, \pi_i = s)\Pr(x_{i+1}, \dots, x_L \mid x_1, \dots, x_i, \pi_i = s)\\ &= \Pr(x_1, \dots, x_i, \pi_i = s)\Pr(x_{i+1}, \dots, x_L \mid \pi_i = s)\,,\end{aligned}$$

using the Markov property in the second equation.

Hence, if we define the *backward-variables*:

$$B(s,i) = \Pr(x_{i+1} \dots x_L \mid \pi_i = s)\,,$$

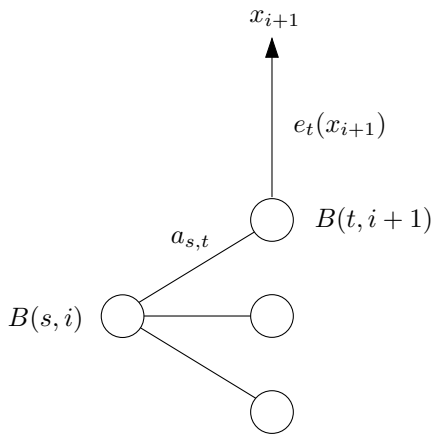
then

$$\begin{aligned}\Pr(x, \pi_i = s) &= \Pr(x_1, \dots, x_i, \pi_i = s)\Pr(x_{i+1}, \dots, x_L \mid \pi_i = s)\\ &= F(s,i)B(s,i)\,.\end{aligned}$$

$B(s,i)$ is the probability that the HMM will generate the sequence $(x_{i+1}, \dots, x_L)$ "when it is started in state π_i", until it reaches state 0. It is computed by the *backward algorithm*.

Input:	HMM $M = (\Sigma, Q, A, e)$ and symbol sequence x
Output:	Probability $\Pr(x)$.
Initialization ($i = L$)*:*	$B(s, L) = a_{s,0}$ for all $s \in Q$.
Recurrence:	
For all $i = L-1 \dots 1$, $s \in Q$:	$B(s,i) = \sum_{t \in Q} a_{st} e_t(x_{i+1}) B(t, i+1)$
Termination ($i = 0$)*:*	$\Pr(x) = \sum_{t \in Q} a_{0,t} e_t(x_1) B(t, 1)$

The reasoning behind the recurrences is similar to the forward algorithm.



Usually one is mainly interested in the backward *variables*, since Pr(x) can be computed by the forward algorithm as well.

9.17 Posterior decoding

The probabilities

$$\Pr(\pi_i = s \mid x), \quad i = 1, \ldots, L, s \in Q$$

are called *posterior probabilities*, because they are determined *after* observing the (random) sequence x. They are conditional probabilities where we have a partial knowledge of the actual outcome of a random experiment.

The sequence of states $\hat{\pi}$ obtained by posterior decoding is defined thus:

$$\hat{\pi}_i = \arg\max_{s \in Q} P(\pi_i = s \mid x)\,.$$

In other words, at every position we choose the most probable state for that position.

Pros and cons of posterior decoding:

- Posterior probabilities provide an alternative means to decode the observed sequence. It is often superior to the Viterbi algorithm, especially if there are *many paths* which have *almost the same probability* as the most probable one.
- Posterior decoding is useful if we are interested in *the state at a specific position i* and not in the whole sequence of states.
- Also the posterior *probabilities* can serve as a *quantitative measurement* of the *confidence* in the predicted states.
- Although the posterior decoding is defined by maximizing the probabilities at each state, *the path as a whole might not be taken very likely* by the HMM. If the transition matrix forbids some transitions (i.e., $a_{st} = 0$), then this decoding may even produce a sequence that is *not a valid path*, because its probability is 0 !

9.18 The design of Hidden Markov Models

How does one come up with an HMM?

First step: Determine its "*topology*", i.e. the number of states and how they are connected via transitions of non-zero probability. – This is kind of an art. We will sketch three examples used in gene prediction and later discuss profile HMMs as another example of a commonly used topology.

Second step: Set the *parameters*, i.e. the transition probabilities a_{st} and the emission probabilities $e_s(x)$.

We consider the second step. Given a set of example sequences. Our goal is to "train" the parameters of the HMM using the example sequences, that is, to set the parameters in such a way that the probability, with which the HMM generates the given example sequences, is maximized.

9.19 Training when the states are known

Let $M = (\Sigma, Q, A, e)$ be an HMM.

Given a list of sequences of symbols $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$ and a list of corresponding paths $\pi^{(1)}, \pi^{(2)}, \ldots, \pi^{(n)}$. (E.g., DNA sequences with annotated CpG-islands.) We consider the probability that the HMM generates them "one after another".

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

$$(A, e) \;=\; \arg\max \left\{ \Pr_{M'}\left((x^{(1)}, \pi^{(1)}), \ldots, (x^{(n)}, \pi^{(n)})\right) \,\middle|\, M' = (\Sigma, Q, A', e') \right\}$$

In other words, we want to determine the *maximum likelihood estimator* (*ML-estimator*) for the parameters (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*.)

Not surprisingly, it turns out (analytically) that the likelihood is maximized by the estimators

$$\tilde{A}_{st} := \frac{\bar{A}_{st}}{\sum_{t'} \bar{A}_{s,t'}} \qquad \text{and} \qquad \tilde{e}_{st} := \frac{\bar{e}_s(t)}{\sum_{t'} \bar{e}_s(t')},$$

where

$$\bar{A}_{st} := \text{Observed number of transitions from state } s \text{ to state } t,$$
$$\bar{e}_s(b) := \text{Observed number of emissions of symbol } b \text{ in state } s.$$

(However this presumes that we have sufficient training data.)

9.20 Training the *fair/loaded* HMM

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:

$\bar{A}_{kl}$	0	F	U
0			
F			
U			

$\rightarrow$

$\tilde{A}_{kl}$	0	F	U
0			
F			
U			

Emissions:

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

$\rightarrow$

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

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:

$\bar{A}_{kl}$	0	F	U
0	0	1	0
F	1	8	1
U	0	1	3

$\rightarrow$

$\tilde{A}_{kl}$	0	F	U
0	0	1	0
F	$\frac{1}{10}$	$\frac{8}{10}$	$\frac{1}{10}$
U	0	$\frac{1}{4}$	$\frac{3}{4}$

Emissions:

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

$\rightarrow$

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

9.21 Pseudocounts

One common problem in training is *overfitting*. For example, if some possible transition (s, t) is never seen in the example data, then we will set $\tilde{a}_{st} = 0$ and the transition is then forbidden in the resulting HMM. Moreover, if a given state s is never seen in the example data, then $\tilde{a}_{st}$ resp. $\tilde{e}_s(b)$ is undefined for all t, b.

To solve this problem, we introduce *pseudocounts* r_{st} and $r_s(b)$ and *add* them to the observed transition resp. emission frequencies.

Small pseudocounts reflect "little pre-knowledge", large ones reflect "more pre-knowledge". The effect of pseudocounts can also be thought of as "smoothing" the model parameters using a background model (i. e., a *prior distribution*).

9.22 Training when the states are unknown

Now we assume we are given a list of sequences of symbols $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$ and we do *not* know the list of corresponding paths $\pi^{(1)}, \pi^{(2)}, \ldots, \pi^{(n)}$, as is usually the case in practice.

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

$$(A, e) \;=\; \arg\max \left\{ \Pr_{M'}\left(x^{(1)}, \ldots, x^{(n)}\right) \,\middle|\, M' = (\Sigma, Q, A', e') \right\}$$

is NP-hard and hence we cannot be solve it exactly in polynomial time.

The Baum-Welch algorithm finds a locally optimal solution to the HMM training problem. It starts from an arbitrary initial estimate for (A, e) and the observed frequencies $\bar{A}$, $\bar{e}$. Then it applies a reestimation step until convergence or timeout.

- In each reestimation step, the forward and backward algorithm is applied (with the current model parameters) to each example sequence.
- Using the posterior state probabilities, one can calculate the expected emission frequencies. (In a loop we add for each training sequence a term to the current $\bar{e}$).
- Using the forward and backward variables, one can also calculate the expected transition frequencies. (In a loop we add for each training sequence a term to $\bar{A}$).
- The new model parameters are calculated as maximum likelihood estimates based on $\bar{A}$, $\bar{e}$.

9.23 Baum-Welch algorithm

Input: HMM $M = (\Sigma, Q, A, e)$,
training data $x^{(1)}, \ldots, x^{(n)}$,
pseudocounts r_{st} and $r_s(b)$ (if desired).

Output: HMM $M' = (\Sigma, Q, A', e')$ with an improved score.

Initialization: Pick some arbitrary model parameters (A, e).
Set some initial "observation frequencies" $(\bar{A}, \bar{e})$.

Reestimation: For each sequence $x^{(j)}$, $j = 1, \ldots, n$:
  Calculate $F^{(j)}$ using the forward algorithm.
  Calculate $B^{(j)}$ using the backward algorithm.
  Update $\bar{A}$ and $\bar{e}$ using the posterior probabilities.
Calculate new model parameters (A, e) from $(\bar{A}, \bar{e})$.
Calculate the new log-likelihood of the model.
Repeat.

Termination: Stop if the change of log-likelihood becomes too small,
or the maximal number of iterations is exceeded.

Updating $\bar{e}$: The probability that the model with parameters (A, e) was in state $\pi_i = s$ when it generated the symbol $x_i^{(j)}$ of $x^{(j)}$ is just the posterior state probability

$$\Pr(\pi_i^{(j)} = s \mid x^{(j)}) = \frac{F^{(j)}(s, i) B^{(j)}(s, i)}{\Pr(x^{(j)})} .$$

Thus we can derive the expected number of times b is output in state s, i.e. $\bar{e}_s(b)$ by

$$\sum_{j=1}^{n} \sum_{i : x_i^{(j)} = b} \frac{F^{(j)}(s, i) B^{(j)}(s, i)}{\Pr(x^{(j)})} .$$

Updating $\bar{A}$: The probability that the model with parameters (A, e) stepped from state $\pi_i = s$ to state $\pi_{i+1} = t$ when it generated the symbols $x_i^{(j)}, x_{i+1}^{(j)}$ of $x^{(j)}$ is

$$\begin{aligned}\Pr(\pi_i = s, \pi_{i+1} = t \mid x) &= \Pr(x_1, \ldots, x_i, \pi_i = s) \cdot \Pr(x_{i+1}, \pi_i = s, \pi_{i+1} = t) \cdot \Pr(x_{i+2}, \ldots, x_L \mid \pi_{i+1} = t) \\ &= \frac{F(s,i) a_{st} e_t(x_{i+1}) B(t, i+1)}{\Pr(x^{(j)})}.\end{aligned}$$

Thus we can derive the expected number of times that a_{st} is used in the training data as follows:

$$\sum_{j=1}^{n} \sum_{i=0}^{|x^{(j)}|} \frac{F(s,i) a_{st} e_t(x_{i+1}) B(t, i+1)}{\Pr(x^{(j)})}.$$

(Here we let $e_t(x_{|x^{(j)}|+1}) := 1$.)

Remark: One can prove that the log-likelihood-score converges to a local maximum using the Baum-Welch-algorithm.

However, this doesn't imply that the parameters converge!

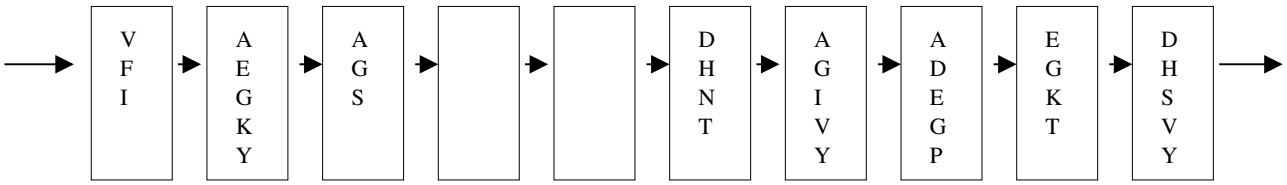
Local maxima can be avoided by considering many different starting points.

Additionally, any standard optimization approaches can also be applied to solve the optimization problem.

9.24 Profile HMMs

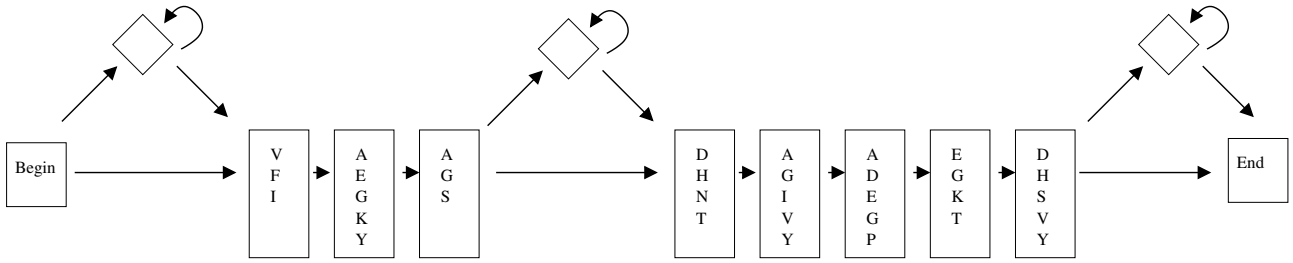
```
HBA_HUMAN     ...VGA--HAGEY...
HBB_HUMAN     ...V----NVDEV...
MYG_PHYCA     ...VEA--DVAGH...
GLB3_CHITP    ...VKG------D...
GLB5_PETMA    ...VYS--TYETS...
LGB2_LUPLU    ...FNA--NIPKH...
GLB1_GLYDI    ...IAGADNGAGV...
"Matches":       ***  *****
```

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



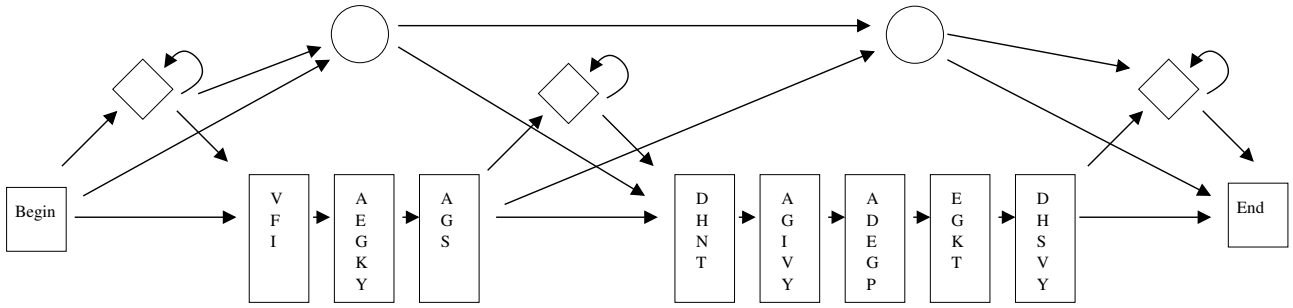
(The listed amino-acids have a higher emission-probability.)

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.

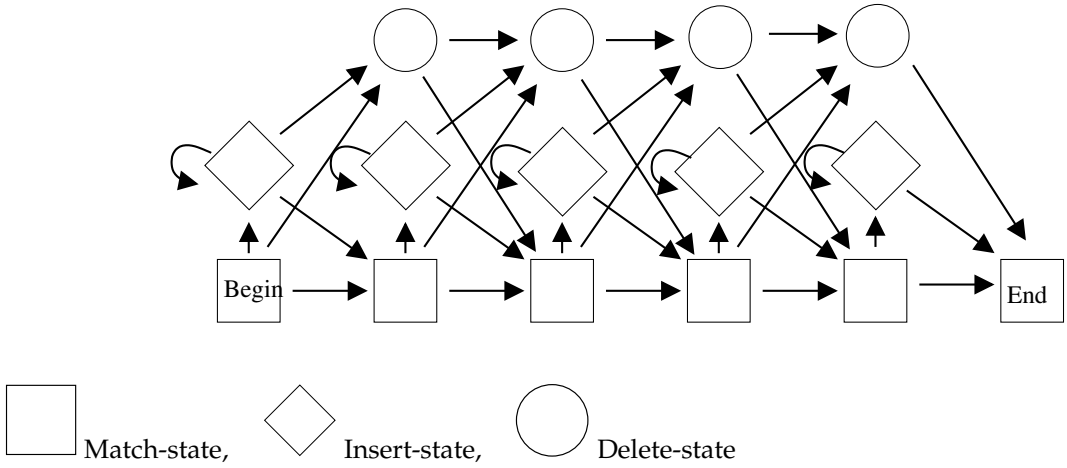
We introduce so-called *delete*-states that are *silent* and do not emit any symbols.



This allows us to model the absence of individual domains.

Silent states can be included into HMMs if we provide some changes to the algorithms. We consider the case where the HMM does not contain cycles involving silent states. Of course, silent states have no factor for the emission probability in the recurrences. In the DP recurrence, for each column the silent states are processed after the emitting states. They receive incoming paths from the same column. We do not describe the details here.

The general topology of a profile HMM is as follows.



9.25 Excursion: Higher-order Markov chains

Remark.
One can also define Markov chains of order k, where the transition probabilities have the form $P(x_{i+1} = t \mid x_i =$

$s_1, \ldots, x_{i-k+1} = s_k$), but these are not easily visualized as graphs and can (at least in principle) be reduced to the order 1 case.

Idea:
Assume that we are given a 2-nd order Markov chain over an alphabet (state set) Σ with transition probabilities

$$a_{s,t,u} = \Pr(x_{i+1} = u \mid x_i = t, x_{i-1} = s)\,.$$

Here the "output" of a state is simply its label.

Then this is equivalent to a 1-st order Markov chain over the alphabet Σ^2 with transition probabilities

$$\tilde{a}_{(s,t),(t',u)} := \begin{cases} a_{s,t,u} & t = t' \\ 0 & \text{otherwise} \end{cases}$$

The "output" of a state (t, u) is its last entry, u.

Begin and end of a sequence can be modeled using a special state 0, as before. The reduction of k-th order Markov chains uses Σ^k in a similar way. (Proof: exercise.) – However, usually it is better to modify the algorithms rather than blow up the state space.