Statistical Computing

Hidden Markov Models for Bioinformatics
- Part IV -

Uwe Menzel, 2011

uwe.menzel@matstat.org

www.matstat.org

Contents Part IV

Forward algorithm

In part 3, we have used the Viterbi algorithm to identify the state path π^* that maximizes the probabilities $P(x,\pi)$ and $P(\pi \mid x)$. Another task is to calculate the probability of the observation, $P(x)$. By taking advantage of the marginal rule, $P(x)$ can be calculated as the sum over all state paths leading to observation x:

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

Many state paths can lead to the same observation, as we have seen for the CpG island example in part 3, giving rise to a high computational load when the above formula is used.

The forward algorithm provides a scheme to calculate $P(x)$ in a recursive manner, similar to the Viterbi algorithm. We define the forward variable $f_k(i-1)$, the probability of the chain up to observation x_{i-1}, ending with state $\pi_{i-1} = k$:

$$f_k(i-1) = P(x_1, x_2, \dots, x_{i-1}, \pi_{i-1} = k)$$

Forward algorithm

In order to obtain the forward variable at position i, we replace the $max_k(\dots)$-operator in the Viterbi algorithm by the sum by $\sum_k(\dots)$.

$$f_l(i) = e_l(x_i) \cdot \sum_k \{f_k(i-1) \cdot a_{kl}\} \qquad i = 1, 2, \dots, L$$

That ensures that we obtain a sum over all state paths leading to the observation $x_1, x_2, \dots, x_i$. We initialise with $f_0(0) = 1$, and $f_k(0) = 0$ for $k \neq 0$:

$$i = 1 \;\; \rightarrow \;\; f_l\,(1) = e_l(x_1) \cdot f_0(0) \cdot a_{0l} = e_l(x_1) \cdot a_{0l}$$

- the index l runs through all states, for example $A^+, C^+, G^+, T^+, A^-, C^-, G^-, T^-$
- x_1 is the first observation in the chain, for instance C (CpG island example)

$$i = 2 \;\; \rightarrow f_l\,(2) = e_l(x_2) \cdot \sum_k \{f_k(1) \cdot a_{kl}\}$$

- both indices k and l run through all states, e.g. $A^+, C^+, G^+, T^+, A^-, C^-, G^-, T^-$
- x_2 is the second observation in the chain, for instance G (CpG island example)
- we continue running through index i ....

Forward algorithm

The last step is for $i = L$:

$$i = L \;\; \rightarrow f_l\,(L) = e_l(x_L) \;\cdot\; \sum_k \{f_k(L-1) \;\cdot\; a_{kl}\}$$

- both indices k and l run through all states, e.g. $A^+, C^+, G^+, T^+, A^-, C^-, G^-, T^-$
- x_L is the last observation in the chain, for instance G

Now, we have the variables $f_l(L)$ for all l, i.e. for all possible states the chain can end in. If it is known that the chain ends after L symbols, we can then calculate the desired $P(x)$ as:

$$P(x) = \sum_k f_k(L)$$

For chains of unknown length (embedded sequence) it might be convenient to introduce an end state E, with transition probabilities a_{kE} (or a_{k0}) to that state. If an end state is included in the model, $P(x)$ is calculated as:

$$P(x) = \sum_k \{f_k(L) \cdot a_{kE}\}$$

Forward algorithm $\quad i = 1 \;\rightarrow\; f_l(1) = e_l(x_1) \cdot a_{0l}$

Recursion: $f_l(i) = e_l(x_i) \cdot \sum_k \{f_k(i-1) \cdot a_{kl}\}$

Observed sequence: **C G C G**

$x_1 = C\,;\, x_2 = G\,;\, x_3 = C\,;\, x_4 = G$

We had $f_0(0) = 1$, and $f_k(0) = 0$ for $k \neq 0$

l	$f_l(1)$	$f_l(1)$
A^+	$e_{A^+}(C) \cdot a_{0A^+}$	0
C^+	$e_{C^+}(C) \cdot a_{0C^+}$	0.5
G^+	$e_{G^+}(C) \cdot a_{0G^+}$	0
T^+	$e_{T^+}(C) \cdot a_{0T^+}$	0
A^-	$e_{A^-}(C) \cdot a_{0A^-}$	0
C^-	$e_{C^-}(C) \cdot a_{0C^-}$	0.5
G^-	$e_{G^-}(C) \cdot a_{0G^-}$	0
T^-	$e_{T^-}(C) \cdot a_{0T^-}$	0

To ease calculations, we set $a_{0C^+} = a_{0C^-} = 0.5$ here (we know that the chain starts with symbol C)

Forward algorithm

$i = 2 \;\rightarrow f_l(2) = e_l(x_2) \cdot \sum_k \{f_k(1) \cdot a_{kl}\}$

Recursion: $f_l(i) = e_l(x_i) \cdot \sum_k \{f_k(i-1) \cdot a_{kl}\}$
Observed sequence: **C G C G**
$x_1 = C\,;\, x_2 = G\,;\, x_3 = C\,;\, x_4 = G$

We have $f_{C^+}(1) = 0.5$, and $f_{C^-}(1) = 0.5$, all other $f_l(1) = 0$

l	$f_l(2)$	$f_l(2)$
A^+	$e_{A^+}(G) \cdot \sum_k \{\dots\}$	0
C^+	$e_{C^+}(G) \cdot \sum_k \{\dots\}$	0
G^+	$e_{G^+}(G) \cdot [f_{C^+}(1) \cdot a_{C^+G^+} + f_{C^-}(1) \cdot a_{C^-G^+}]$	$0.5 \cdot (0.26 + 0.0025) = 0.13125$
T^+	$e_{T^+}(G) \cdot \sum_k \{\dots\}$	0
A^-	$e_{A^-}(G) \cdot \sum_k \{\dots\}$	0
C^-	$e_{C^-}(G) \cdot \sum_k \{\dots\}$	0
G^-	$e_{G^-}(G) \cdot [f_{C^+}(1) \cdot a_{C^+G^-} + f_{C^-}(1) \cdot a_{C^-G^-}]$	$0.5 \cdot (0.0125 + 0.077) = 0.04475$
T^-	$e_{T^-}(G) \cdot \sum_k \{\dots\}$	0

Forward algorithm $i = 3 \rightarrow f_l(3) = e_l(x_3) \cdot \sum_k \{f_k(2) \cdot a_{kl}\}$

Recursion: $f_l(i) = e_l(x_i) \cdot \sum_k \{f_k(i-1) \cdot a_{kl}\}$

Observed sequence: **C G C G**

$x_1 = C\ ;\ x_2 = G\ ;\ x_3 = C\ ;\ x_4 = G$

We had $f_{G^+}(2) = 0.13125$, and $f_{G^-}(2) = 0.04475$, all other $f_l(2) = 0$

l	$f_l(3)$	$f_l(3)$
A^+	$e_{A^+}(C) \cdot \sum_k \{\ldots\}$	0
C^+	$e_{C^+}(C) \cdot [f_{G^+}(2) \cdot a_{G^+C^+} + f_{G^-}(2) \cdot a_{G^-C^+}]$	$(0.13125 \cdot 0.322 + 0.04475 \cdot 0.0025) = 0.04237$
G^+	$e_{G^+}(C) \cdot \sum_k \{\ldots\}$	0
T^+	$e_{T^+}(C) \cdot \sum_k \{\ldots\}$	0
A^-	$e_{A^-}(C) \cdot \sum_k \{\ldots\}$	0
C^-	$e_{C^-}(C) \cdot [f_{G^+}(2) \cdot a_{G^+C^-} + f_{G^-}(2) \cdot a_{G^-C^-}]$	$(0.13125 \cdot 0.0125 + 0.04475 \cdot 0.244) = 0.01256$
G^-	$e_{G^-}(C) \cdot \sum_k \{\ldots\}$	0
T^-	$e_{T^-}(C) \cdot \sum_k \{\ldots\}$	0

Forward algorithm

$i = 4 \;\rightarrow f_l(4) = e_l(x_4) \cdot \sum_k \{f_k(3) \cdot a_{kl}\}$

Recursion: $f_l(i) = e_l(x_i) \cdot \sum_k \{f_k(i-1) \cdot a_{kl}\}$

Observed sequence: **C G C G**

$x_1 = C\,;\, x_2 = G\,;\, x_3 = C\,;\, x_4 = G$

We had $f_{C^+}(3) = 0.04237$, and $f_{C^-}(3) = 0.01256$, all other $f_l(3) = 0$

l	$f_l(4)$	$f_l(4)$
A^+	$e_{A^+}(G) \cdot \sum_k \{\dots\}$	0
C^+	$e_{C^+}(G) \cdot \sum_k \{\dots\}$	0
G^+	$e_{G^+}(G) \cdot [f_{C^+}(3) \cdot a_{C^+G^+} + f_{C^-}(3) \cdot a_{C^-G^+}]$	$(0.04237 \cdot 0.26 + 0.01256 \cdot 0.0025) = 0.01104$
T^+	$e_{T^+}(G) \cdot \sum_k \{\dots\}$	0
A^-	$e_{A^-}(G) \cdot \sum_k \{\dots\}$	0
C^-	$e_{C^-}(G) \cdot \sum_k \{\dots\}$	0
G^-	$e_{G^-}(G) \cdot [f_{C^+}(3) \cdot a_{C^+G^-} + f_{C^-}(3) \cdot a_{C^-G^-}]$	$(0.04237 \cdot 0.0125 + 0.01256 \cdot 0.077) = 0.001497$
T^-	$e_{T^-}(G) \cdot \sum_k \{\dots\}$	0

Forward algorithm, results

Recursion: $f_l(i) = e_l(x_i) \cdot \sum_k \{f_k(i-1) \cdot a_{kl}\}$
Observed sequence: **C G C G**
$x_1 = C\ ; x_2 = G\ ; x_3 = C\ ; x_4 = G$

We had $f_{G^+}(4) = 0.01104$, and $f_{G^-}(4) = 0.001497$, all other $f_l(4) = 0$

The chain ends after 4 symbols, so that the desired $P(x)$ is:

$$P(x) = \sum_k f_k(L)$$

$$P(x) = f_{G^+}(4) + f_{C^-}(4) = 0.01104 + 0.001497 \approx 0.01254$$

Note: For long sequences, it is suggested to carry out the calculations in log-space, in order to avoid underflow. Products are replaced by sums in that case.

Forward algorithm, results

Recursion: $f_l(i) = e_l(x_i) \cdot \sum_k \{f_k(i-1) \cdot a_{kl}\}$

Observed sequence: **C G C G**

$x_1 = C\,;\, x_2 = G\,;\, x_3 = C\,;\, x_4 = G$

l	$f_l(1)$	$f_l(2)$	$f_l(3)$	$f_l(4)$
A^+	0	0	0	0
C^+	0.5	0	0.04237	0
G^+	0	0.13125	0	0.01104
T^+	0	0	0	0
A^-	0	0	0	0
C^-	0.5	0	0.01256	0
G^-	0	0.04475	0	0.001497
T^-	0	0	0	0

Forward algorithm

C:\Users\Uwe\Desktop\TALKS_POSTERS\LECTURES\HMM-Talk am HKI\HMM_forward_CGCG.R - R Editor

```
library(HMM)
states = c("A+", "C+", "G+", "T+", "A-", "C-", "G-", "T-")
symbols = c("A", "C", "G", "T")
trans_prob = get(load("trans_prob_HMM.RData"))
emission_prob = get(load("emission_prob_HMM.RData"))
start_prob = c(0, 0.5, 0,0,0,0.5,0,0)
names(start_prob) = c("A+", "C+", "G+", "T+", "A-", "C-", "G-", "T-")
hmm = initHMM(states, symbols, startProbs = start_prob,
              transProbs = trans_prob, emissionProbs = emission_prob)

observation = c("C", "G", "C", "G")    # observation

log_fward = forward(hmm, observation)  # forward algorithm
exp(log_fward)
#      A+ 0.0 0.00000 0.00000000 0.000000000
#      C+ 0.5 0.00000 0.04237438 0.000000000
#      G+ 0.0 0.13125 0.00000000 0.011048737
#      T+ 0.0 0.00000 0.00000000 0.000000000
#      A- 0.0 0.00000 0.00000000 0.000000000
#      C- 0.5 0.00000 0.01255962 0.000000000
#      G- 0.0 0.04475 0.00000000 0.001496771
#      T- 0.0 0.00000 0.00000000 0.000000000
```

Forward algorithm

Uwe Menzel, 2011

Posterior probabilities

So far we have used the Viterbi algorithm to identify the state path π^* that maximizes the probabilities $P(x, \pi)$ and $P(\pi \mid x)$, respectively.
Another decoding approach is posterior decoding. For posterior decoding, we calculate the probability $P(\pi_i = k \mid x)$, i.e. the probability that the state π_i is k given the observed sequence, for all positions i and all states k. We then use the expression

$$\hat{\pi}_i = argmax_k \, P(\pi_i = k \mid x)$$

to infer the most likely state for each position i. By doing so, we focus our attention on particular states π_i, rather than on the state path as a whole. The state sequence that emerges from stringing together all the $\hat{\pi}_i$ is not the same as π^*. It might even happen that two states π_i and π_{i+1} are concatenated to become neighbors in such a state sequence, although a transit between them is forbidden, because the transition probability $a_{\pi_i \pi_{i+1}}$ is zero. Posterior decoding is especially useful when many state paths approximately account for the same probability of the chain because posterior decoding does not exclusively zoom in on the single most probable path as calculated by the Viterbi algorithm.

Posterior probabilities

The posterior probabilities can be calculated by making use of the forward variables introduced above. Using the definition of conditional probability, we can write:

$$\begin{aligned} P(x, \pi_i = k) &= P(x_1, x_2, \ldots, x_i, x_{i+1}, \ldots, x_L, \pi_i = k) \\ &= P(x_1, x_2, \ldots, x_i, \pi_i = k) \cdot P(x_{i+1}, \ldots, x_L \mid x_1, x_2, \ldots, x_i, \pi_i = k) \end{aligned}$$

The Markov property says that the observations $x_{i+1}, \ldots, x_L$ can only depend on the state π_i, but not on $x_1, x_2, \ldots, x_i$, so that the latter simplifies to:

$$P(x, \pi_i = k) = \underbrace{P(x_1, x_2, \ldots, x_i, \pi_i = k)}_{f_k(i) = P(x_1, x_2, \ldots, x_i, \pi_i = k)} \cdot \underbrace{P(x_{i+1}, \ldots, x_L \mid \pi_i = k)}_{b_k(i)}$$

The first factor is the forward variable, which we can calculate using the recursive algorithm presented above. The second factor is the so-called backward variable, $b_k(i)$. The backward variable can also be calculated using a recursive procedure, the backward algorithm →

Backward algorithm

The backward algorithm starts at the **last** symbol in the chain. We define the variable $b_k(i)$, the probability of the chain up to observation x_{i+1}, counted from the end of the chain, ending in state $\pi_i = k$:

$$b_k(i) = P(x_{i+1}, x_{i+2}, \dots, x_L \mid \pi_i = k)$$

The $b_k(i)$ can be calculated recursively (L = length of the chain):

$$b_k(i) = \sum_n e_n(x_{i+1}) \cdot b_n(i+1) \cdot a_{kn} \qquad i = L-1, L-2, \dots, 1$$

The indices n and k run through all states, e.g. $A^+, C^+, G^+, T^+, A^-, C^-, G^-, T^-$

For $i = L$ we initialise: $b_k(L) = a_{k0}$ for all states k.

The backward algorithm also delivers $P(x)$, by

$$P(x) = \sum_n e_n(x_1) \cdot b_n(1) \cdot a_{0n}$$

Backward algorithm

Termination: $$P(x) = \sum_n e_n(x_1) \cdot b_n(1) \cdot a_{0n}$$

For the example already discussed above, we had $x_1 = C$, $a_{0C^+} = 0.5$ and $a_{0C^-} = 0.5$, so that we the sum reduces to two terms:

$$P(x) = e_{C^+}(C) \cdot b_{C^+}(1) \cdot a_{0C^+} + e_{C^-}(C) \cdot b_{C^-}(1) \cdot a_{0C^-}$$

$$P(x) = 1 \cdot b_{C^+}(1) \cdot 0.5 + 1 \cdot b_{C^-}(1) \cdot 0.5 \approx 0.01254$$

C:\Users\Uwe\Desktop\TALKS_POSTERS\LECTURES\HMM-Talk am HKI\HMM_forward_CGCG.R ...

```
log_bward = backward(hmm, observation) # backward algorithm
exp(log_bward)

# states           1          2      3 4
#     A+ 0.036190184 0.07184375 0.4175 1
#     C+ 0.023323066 0.09636875 0.2725 1
#     G+ 0.031841986 0.08873875 0.3685 1
#     T+ 0.032640634 0.09282625 0.3775 1
#     A- 0.005884195 0.01681975 0.2845 1
#     C- 0.001767949 0.02413375 0.0795 1
#     G- 0.006145226 0.02007925 0.2975 1
#     T- 0.006024750 0.01856875 0.2915 1
```

$b_{C^+}(1) = 0.023323$

$b_{C^-}(1) = 0.0017679$

Posterior decoding

$$P(x, \pi_i = k) = \underbrace{P(x_1, x_2, \dots, x_i, \pi_i = k)}_{f_k(i) = P(x_1, x_2, \dots, x_i, \pi_i = k)} \cdot \underbrace{P(x_{i+1}, \dots, x_L \mid \pi_i = k)}_{b_k(i)}$$

Having all the $f_k(i)$ and $b_k(i)$, we can calculate $P(x, \pi_i = k)$ for all i and k:

$$P(x, \pi_i = k) = f_k(i) \cdot b_k(i)$$

We were actually seeking $P(\pi_i = k \mid x)$. Using the definition of conditional probability, we can write

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

so that we finally get:

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

$P(x)$ can be taken from the forward- or the backward calculation.

Posterior decoding

C:\Users\Uwe\Desktop\TALKS_POSTERS\LECTURES\HMM-Talk am HKI\HMM_forward_CGCG.R - R Editor

```
library(HMM)
states = c("A+", "C+", "G+", "T+", "A-", "C-", "G-", "T-")
symbols = c("A", "C", "G", "T")
trans_prob = get(load("trans_prob_HMM.RData"))
emission_prob = get(load("emission_prob_HMM.RData"))
start_prob = c(0, 0.5, 0,0,0,0.5,0,0)
hmm = initHMM(states, symbols, startProbs = start_prob,
              transProbs = trans_prob, emissionProbs = emission_prob)
observation = c("C", "G", "C", "G")    # observation
posterior = posterior(hmm, observation)
posterior
# states        |1          2          3          4
#     A+ 0.00000000 0.00000000 0.00000000 0.0000000
#     C+ 0.92953856 0.00000000 0.92041054 0.0000000
#     G+ 0.00000000 0.92837703 0.00000000 0.8806927
#     T+ 0.00000000 0.00000000 0.00000000 0.0000000
#     A- 0.00000000 0.00000000 0.00000000 0.0000000
#     C- 0.07046144 0.00000000 0.07958946 0.0000000
#     G- 0.00000000 0.07162297 0.00000000 0.1193073
#     T- 0.00000000 0.00000000 0.00000000 0.0000000
```

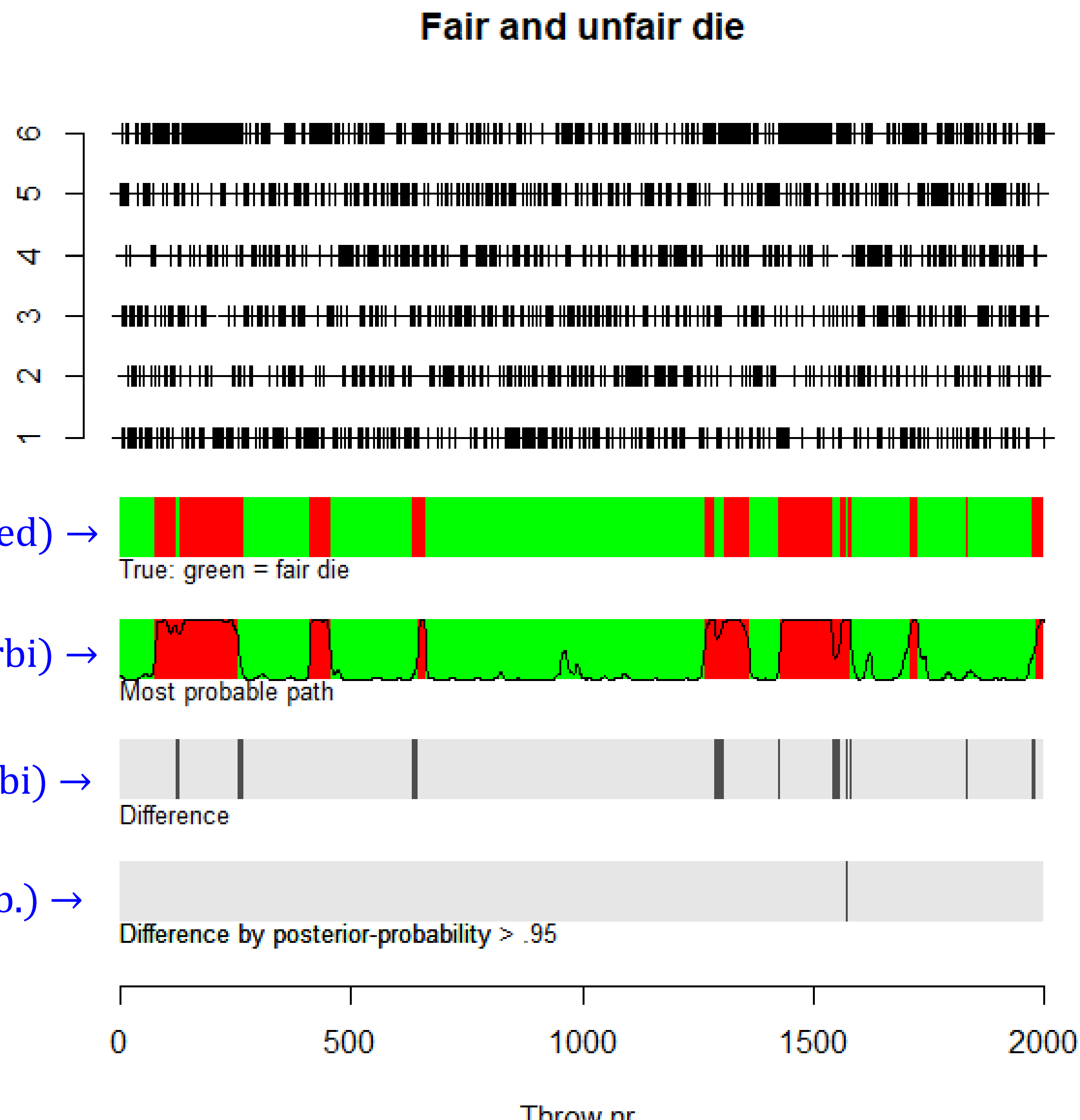
Uwe Menzel, 2011

Here, we have $\hat{\pi}_i = argmax_k \; P(\pi_i = k \mid x) = \{C^+G^+C^+G^+\}$, same as for Viterbi.

Comparison of the most probable path (Viterbi) and the path identified by posterior decoding (forward-backward algorithm)

The library HMM provides a function for demonstration of the casino example:

```
library(HMM)
dishonestCasino()
```



the true states (simulated) →

colors: most probable path (Viterbi) →
black line: posterior probability

difference true ⇄ Viterbi) →

difference true ⇄ posterior prob.) →