It is easy to see that the *Zuker* algorithm excludes pseudoknots. The structure with lowest free energy for the subsequence $S[i \dots j]$ is only allowed to contain non-crossing interactions within this interval. Otherwise, the dynamic programming principle is violated and efficient computation becomes impossible. Pseudoknot interaction happens in a crossing fashion and therefore dynamic programming algorithms for RNA secondary structure prediction neglect them.

3.7 Viterbi Algorithm

3.7.1 Problem Restated

Let $M = (Q, \Sigma, q_0, T, E)$ be a Hidden Markov model with set of states Q, alphabet Σ of emitted characters, start state q_0, state transition probabilities $T(p, q)$, and character emission probabilities $E(q, x)$. Given an observed sequence $S \in \Sigma^n$, the Viterbi algorithm computes the most probable state sequence $W \in Q^n$ that might have emitted string S.

HMM DECODING
Given string $S \in \Sigma^n$, compute $W \in Q^n$ with maximal value $P(W, S)$.

3.7.2 Parameterization and Conditioning

For each $q \in Q$ and $i = 1, \dots, n$ define:

$$\mu_i(q) = \max_{W \in Q^{i-1}} P(Wq, S[1 \dots i]). \tag{3.21}$$

Here, conditioning is on the last used state q in a state sequence that emitted prefix $S[1 \dots i]$.

3.7.3 Bellman Principle

... is obviously fulfilled.

3.7.4 Recursive Computation

$$\begin{aligned} \mu_1(q) &= T(q_0, q) E(q, S(1)) \\ \mu_{i+1}(q) &= E(q, S(i+1)) \max_{r \in Q} \{\mu_i(r) T(r, q)\}. \end{aligned} \tag{3.22}$$

Having computed these values the original task is solved by:

$$\mu_S = \max_{q \in Q} \mu_n(q). \tag{3.23}$$

3.7.5 Counting and Overall Complexity

Obviously, $O(n|Q|^2)$ steps are required to compute all these values $\mu_i(q)$.

3.8 Baum-Welch Algorithm

3.8.1 Problem Motivation

Next we discuss an often used way to fit parameters of a Hidden Markov model M to observations consisting of strings $S_1, \ldots, S_m$. The Viterbi algorithm computes the most likely state strings $W_1, \ldots, W_m$. Afterwards, these together with $S_1, \ldots, S_m$ are used to count transition and emission frequencies which are used as fresh values for an updated model M^*. Though being intuitive, the disadvantage of this approach is that state strings are used that maximize joint probability, and working with maxima is notoriously difficult in probability theory. It is much better to work with expectations. In our context this means that we use all state sequences to update transition and emission probabilities, however each state sequence is weighted with its probability of occurrence. Thus, plausible sequences are preferred via their probability of occurrence. For example, we thus have to compute the expected transition frequency between any two states, the expectation taken over all state sequences. If this is done, one can indeed formally prove that for the updated model M^* likelihood of emitting $S_1, \ldots, S_m$ is improved compared to model M. This is an example of a more fundamental algorithm, called *expectation maximization* (EM) algorithm.

3.8.2 A Couple of Important 'Variables'

In probability theory, a function defined on a probability space usually is called a 'variable'. Given a Hidden Markov model M, the following variables play an important role. For a fixed emitted string $S \in \Sigma^n, 1 \leq i \leq n$ and $1 \leq j < n$ we define them as follows:

- *Forward variable* $\alpha_i(q)$ as the probability that M generates a state sequence with i^{th} state q and emits a symbol sequence with prefix $S[1 \ldots i]$.
- *Backward variable* $\beta_i(q)$ as the probability that M emits a symbol sequence with suffix $S[i+1 \ldots n]$ provided it generated a state sequence with i^{th} state q.
- *Transition variable* $\eta_i(q, r)$ as the probability that M generates a state sequence with i^{th} state q and $(i+1)^{\text{th}}$ state r provided it emitted symbol sequence S.
- *State variable* $\gamma_i(q)$ as the probability that M generates a state sequence with i^{th} state q provided it emitted symbol sequence S.

3.8.3 Computing Forward Variables

The forward variables are computed in $O(|S||Q|^2)$ steps as follows:

$$\begin{aligned}\alpha_1(q) &= T(q_0, q)E(q, S(1)) \\ \alpha_{i+1}(q) &= E(q, S(i+1)) \sum_{r \in Q} \alpha_i(r)T(r, q).\end{aligned} \tag{3.24}$$

As an application of forward variables, we see that emission probabilities $P(S)$ of string S of length n (original definition required an exponential summation) can be computed efficiently:

$$P(S) = \sum_{q \in Q} \alpha_n(q). \tag{3.25}$$

3.8.4 Computing Backward Variables

The backward variables are computed similarly in $O(|S||Q|^2)$ steps as follows:

$$\begin{aligned}\beta_n(q) &= 1 \\ \beta_{i-1}(q) &= \sum_{r \in Q} T(q, r)E(r, S(i))\beta_i(r).\end{aligned} \tag{3.26}$$

Exercise

3.6. Express $P(S)$ using backward variable instead of forward variable as in Sect. 3.8.3.

3.8.5 Computing Transition Variables

Now we can obtain the transition variables on basis of the following equation. Its left-hand side is the probability of observing string S and having states q and r at positions j and $j+1$, whereas the right-hand side decomposes this event into observing prefix $S[1 \ldots j]$ and having state q at position j, then switching to state r at position $j+1$ and emitting $S(j+1)$, and finally observing suffix $S[j+1 \ldots n]$ after starting with state r at position $j+1$.

$$\eta_j(q, r)P(S) = \alpha_j(q)T(q, r)E(r, S(j+1))\beta_{j+1}(r) \tag{3.27}$$

3.8.6 Computing State Variables

$$\begin{aligned}\gamma_1(q) &= T(q_0, q) \\ \gamma_{i+1}(q) &= \sum_{r \in Q} \eta_i(r, q)\end{aligned} \tag{3.28}$$

Exercise

3.7. Explain these equations.

3.8.7 Model Improvement

Let string S of length n be observed as the emitted string. Based on S we compute the variables as above and use them to update transition and emission probabilities as follows, for states q and r different from initial state q_0:

$$\begin{aligned} T^{\text{update}}(q,r) &= \frac{\sum_{j=1}^{n-1} \eta_j(q,r)}{\sum_{j=1}^{n-1} \gamma_j(q)} \\ T^{\text{update}}(q_0,q) &= \gamma_1(q) \\ E^{\text{update}}(q,x) &= \frac{\sum_{i\le n, S(i)=x} \gamma_i(q)}{\sum_{i\le n} \gamma_i(q)} \end{aligned} \tag{3.29}$$

It can be shown (see [4]) that this update is a particular case of expectation maximization, and as such indeed leads to improvement of model likelihood. Iterated application thus approaches a local maximum of the model likelihood function.

3.9 Expressiveness of Viterbi Equations

The Viterbi algorithm was a simple application of the general principle of dynamic programming. It is interesting to see that often applications of dynamic programming may be reinterpreted as Viterbi equations for a suitably designed Hidden Markov model. This shows, in a certain sense, a certain generality of the Hidden Markov approach. We illustrate this by showing how affine gap alignment treated in Sect. 3.3 may be reinterpreted as Viterbi algorithms of a suitably designed Hidden Markov model (we follow the treatment in [27]).

The application of the gap alignment equations described in Sect. 3.3 may be visualized by the graph shown in Fig. 3.4 containing start node q_0, and three further nodes called 'a', 'b', and 'c' that correspond to the three functions optimized. Index pair (i,j) maintains which one of the characters of the strings S and T, namely $S(i)$ and $T(j)$, are the actual characters that are next to be aligned or aligned with a spacing symbol. Each node 'a', 'b', and 'c' contains information on how these indices are to be updated whenever the corresponding node is visited. Links contain scores that are to be added onto a growing score whenever the corresponding link is traversed. Note that there are no links between nodes 'b' and 'c'; this expresses our convention that any two b- and c-blocks must be separated by at least one a-block.

This graph gives rise to a similar structure of a Hidden Markov model (HMM) designed for the emission of an alignment (Fig. 3.5). First, the proposed HMM has states q_0, A, B, and C. State A emits pairs (x,y), state B emits pairs $(-,y)$, and state C emits pairs $(x,-)$, with characters x and y from alphabet Σ. Correspondingly, emission probabilities are called $P(x,y)$, $P(x)$, and