3.6 Multiple alignment using hidden Markov models

3.6.1 Finite Markov chains and hidden Markov models

Finite Markov Chains. A discrete-time finite Markov chain process is given by a finite set of "states" labeled \( \{E_1, E_2, \cdots, E_s\} \) and a transition probability matrix \( P_{s \times s} = (p_{ij}) \). At each of the unit time points \( t = 0, 1, 2, \cdots \), the Markov chain process occupies one of these states. In each time step \( (i \text{ to } i+1) \), the process either stays in the same state \( E_i \) with probability \( p_{ii} \) or moves to some other state \( E_j \) with probability \( p_{ij} \).

A finite Markov chain has the following distinguishing properties:

- **The memoryless property.** If at some time \( i \) the process is in the state \( E_i \), the probability that one time unit later it is in state \( E_j \) depends only on \( E_j \), and not on the past history of the states it was in before time \( t \).

- **The time homogeneity property.** Given that at time \( i \), the process is in state \( E \), the probability that one time unit later, it is in state \( E' \) is independent of time \( i \).

We often represent a Markov chain as a directed graph by identifying the states with nodes and the transition probabilities with directed weighted edges. Consider, for example, the Markov chain with states \( E_1 \) and \( E_2 \) and with probability transition matrix

\[
\begin{bmatrix}
0.9 & 0.1 \\
0.7 & 0.3
\end{bmatrix}
\]

This Markov chain is represented by the following graph.

![Graph of a Markov chain with states S1 and S2, transitions 0.9 from S1 to S1, 0.7 from S1 to S2, 0.1 from S2 to S1, and 0.3 from S2 to S2.]
Hidden Markov models (HMMs). An HMM is a finite Markov chain with some extra features. The main addition is that when a state is visited, the state “emits” a letter from a fixed alphabet with a time-independent, but state-dependent probability.

When the HMM runs, there is, first, a sequence of states visited, denoted by

\[ Q : q_1, q_2, q_3, \ldots, \]

and second, a sequence of emitted letters, denoted by

\[ O : o_1, o_2, o_3, \ldots. \]

This two-step process can be visualized as follows

\[ \begin{array}{cccc}
  o_1 & o_2 & o_3 & \cdots \\
  \uparrow & \uparrow & \uparrow & \cdots \\
  \text{initial } q_1 & \rightarrow q_2 & \rightarrow q_3 & \rightarrow \cdots
\end{array} \]

We call \( Q \) “the state sequence” and \( O \) “the observed sequence”. Often we know the sequence \( O \), but do not know \( Q \), which is ‘hidden’.
3.6. MULTIPLE ALIGNMENT USING HIDDEN MARKOV MODELS

Formally, an HMM consists of the following five components:

(1) A set of states \( \{E_1, E_2, \cdots, E_s\} \);
(2) An alphabet of emitted letters \( A = \{a_i|1 \leq i \leq n\} \);
(3) The state transition probability matrix \( P_{s \times s} = (p_{ij}) \);
(4) The emission probability matrix \( B_{s \times n} = (b_i(a_j)) \), where \( b_i(a_j) \) is the probability that state \( s_i \) emits letter \( a_j \);
(5) An initial distribution vector \( \pi = (\pi_i | 1 \leq i \leq s), \pi_i = P[q_1 = S_i] \).

Components (1) and (2) describe the structure of the model, and 3-5 describe the parameters. It is convenient to let \( \lambda = (P, B, \pi) \) represent the full set of parameters.

There are three questions that arise frequently in HMM theory. Given an observed sequence \( \mathcal{O} : o_1, o_2, \cdots, o_t \), one may ask:

(1). Given the parameters \( \lambda = (P, B, \pi) \), what is \( P[\mathcal{O}|\lambda] \), the probability of \( \mathcal{O} \) as an observed output?

(2). What is the hidden sequence \( \mathcal{Q} : q_1, q_2, \cdots, q_t \) of states that is most likely to have occurred, given \( \mathcal{O} \)? That is, we need to find \( \text{argmax}_Q P[\mathcal{Q}|\mathcal{O}] \).

(3). Assuming a fixed topology of the model, what are the parameters \( \lambda = (P, B, \pi) \) that maximize \( P[\mathcal{O}|\lambda] \)?
Consider the Markov chain given by the following graph

Let $A = \{1, 2\}$. State $S_1$ emits a 1 or 2 with equal probability $1/2$; state $S_2$ emits a 1 with probability $1/4$ and a 2 with probability $3/4$.

What sequence of states $Q : q_1, q_2, q_3$ has the highest probability $P[Q | O]$ for $O : 2, 2, 2$ assume the uniform initial state distribution?

There are eight possibilities for $Q$. They are

- $S_1, S_1, S_1$
- $S_1, S_1, S_2$
- $S_1, S_2, S_1$
- $S_1, S_2, S_2$
- $S_2, S_1, S_1$
- $S_2, S_1, S_2$
- $S_2, S_2, S_1$
- $S_2, S_2, S_2$

The answer to above question is $S_2, S_1, S_1$. This can be seen from the following formulas $P[Q | O] = P[Q \land O] / P[O]$ and

\[
P[Q = (S_2, S_1, S_1) \land O] = P[\text{Initial } S_2] \cdot P[S_2 \text{ emits 2}] \times
\]
\[
P[S_1 | S_2] \cdot P[S_1 \text{ emits 2}] \cdot P[\text{it stays at } S_1] \cdot P[S_1 \text{ emits 2}]
\]
\[
= 0.5 \cdot \left(\frac{3}{4}\right) \cdot 0.7 \cdot \left(\frac{1}{2}\right) \cdot 0.9 \cdot \left(\frac{1}{2}\right)
\]
\[
\approx 0.0591
\]

What is $P[O]$? 0.1867
Example 2. (1) Consider the following HMM that has

- State set \( \{E_1, E_2, E_3, E_4, E_5, E_6\} \).
- Emission alphabet \( \sigma = \{\#, a, g, c, t, \$\} \).
- State transition probability matrix
  \[
  P_{6 \times 6} = \begin{pmatrix}
  0 & 1 & 0 & 0 & 0 & 0 \\
  0 & 0 & 1 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 0 & 0 & 1 \\
  0 & 0 & 0 & 0 & 0 & 0 \\
  \end{pmatrix}
  \]
  where the \( i \)th row and column correspond to state \( E_i \).
- Emission probability matrix
  \[
  B_{6 \times 6} = \begin{pmatrix}
  1 & 0.25 & 0.25 & 0.25 & 0.25 \\
  0.25 & 0.25 & 0.25 & 0.25 & 0.25 \\
  0.25 & 0.25 & 0.25 & 0.25 & 0.25 \\
  0.25 & 0.25 & 0.25 & 0.25 & 0.25 \\
  0.25 & 0.25 & 0.25 & 0.25 & 0.25 \\
  \end{pmatrix}
  \]
  where the \( i \)th row corresponds to state \( E_i \), and the first and last column correspond to the letter \( \# \) and \( \$ \) respectively.
- Initial state distribution \( \pi = (1, 0, 0, 0, 0, 0) \).

(2). An HMM that specifies a single sequence of length 5.

(3). An HMM that specifies a set of sequences of the form \( t \ast g \ast \ast \ast t \).
3.6.2 The forward and backward algorithms

Given the parameters $\lambda = (M, B, \pi)$, it is easy to calculate $P[\mathcal{O}|\lambda]$ for $\mathcal{O} : o_1, o_2, \cdots, o_T$. The naive way is based on the following formula

$$P[\mathcal{O}|\lambda] = \sum_{\mathcal{Q}} P[\mathcal{O}|\mathcal{Q}] \cdot P[\mathcal{Q}] = \sum_{\mathcal{Q}} P[\mathcal{Q} \land \mathcal{O}].$$

This calculation involves the sum of $N^T$ multiplications. This is not feasible when $T$ is large. Noticing that when we calculate different terms in above formula, we compute some intermediate results repeatedly, we conclude that the dynamic programming approach can be used for the purpose.

Set

$$\alpha(t, S_i) = P[o_1, o_2, \cdots, o_t \land q_t = S_i], \quad t \leq T.$$

Once we know $\alpha(T, S_i)$ for all $i$, then we can compute $P[\mathcal{O}|\lambda]$ as

$$P[\mathcal{O}|\lambda] = \sum_{S_i} \alpha(T, S_i).$$

The \textit{forward algorithm} computes $\alpha(t, i)$ using the following recurrence relations:

(i) Basis condition: $\alpha(1, S_i) = \pi_i b_i(o_1)$.

(ii) The induction step: $\alpha(t + 1, S_i) = \sum_{S_j} \alpha(t, S_j) p_{ji} b_i(o_{t+1})$.

It is not difficult to see that the forward algorithm takes $O(NT^2)$ time.
3.6. MULTIPLE ALIGNMENT USING HIDDEN MARKOV MODELS

Example Consider the previous HMMs with uniform state distribution $\pi = (1/2, 1/2)$. Given $O : 2, 2, 2$ and $T = 3$, the tabular computation of $\alpha(t, S_i)$’s is

<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>$S_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t=1$</td>
<td>0.25</td>
<td>0.375</td>
</tr>
<tr>
<td>$t=2$</td>
<td>0.244</td>
<td>0.103</td>
</tr>
<tr>
<td>$t=3$</td>
<td>0.146</td>
<td>0.041</td>
</tr>
</tbody>
</table>

In the above, we compute $\alpha(1, S_i), \alpha(2, S_i), \ldots, \alpha(T, S_i)$ successively, that is we calculated forward in time. We can also calculate backward in time. Specifically, we calculate the probability $\beta(t, i)$, defined by

$$\beta(t, S_i) = P[o_{t+1}, o_{t+2}, \ldots, o_T | q_t = S_i], \quad t \leq T - 1,$$

starting with $t = T - 1$, then $T - 2$ and so on. Initially,

$$\beta(T, S_i) = 1$$

and for $t \leq T$,

$$\beta(t - 1, S_i) = \sum_{S_j} p_{ij} b_j(o_t) \beta(t, j).$$

Finally, we have that

$$P[O | \lambda] = \sum_{S_i} \pi_i b_i(o_1) \beta(1, S_i).$$

Example (Con’t). Given $O : 2, 2, 2$, the tabular computation of $\beta(t, S_i)$ is

<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>$S_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t=3$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$t=2$</td>
<td>0.525</td>
<td>0.575</td>
</tr>
<tr>
<td>$t=1$</td>
<td>0.279</td>
<td>0.313</td>
</tr>
</tbody>
</table>
3.6.3 The Viterbi algorithm

The Viterbi algorithm is used to find a state sequence \( Q : q_1, q_2, \ldots, q_T \) that has the highest conditional probability given an observed sequence \( O : o_1, o_2, \ldots, o_T \). That is, \( Q = \arg\max_{Q'} P[Q'|O] \).

Since \( P[Q'|O] = \frac{P[Q' \land O]}{P[O]} \) and the denominator on the right-hand side does not depend on \( Q' \),
\[
\arg\max_{Q'} P[Q'|O] = \arg\max_{Q'} P[Q' \land O].
\]

First, define
\[
\delta(1, S_i) = P[q_1 = S_i \land o_1] = P[q_1 = S_i] \cdot P[S_i \text{ emits } o_1] = \pi_i b_i(o_1),
\]
\[
\delta(t, S_i) = \max_{q_1, q_2, \ldots, q_{t-1}} P[q_1, q_2, \ldots, q_{t-1}, q_t = S_i \land o_1, o_2, \ldots, o_t], \forall t, i.
\]
It is easy to see \( \delta(t, S_i) \) is the maximum probability of all ways to end in state \( S_i \) at time \( t \) and have observed sequence \( o_1, o_2, \ldots, o_t \). Obviously,
\[
\max_i \delta(T, i) = \max_{Q'} P[Q' \land O].
\]

The Viterbi algorithm is another dynamic programming algorithm. It is divided into two parts. \textbf{It first finds} \( \max_{Q'} P[Q' \land O] \), \textbf{and then “backtrack”} to find an \( Q \) that realizes this maximum.
3.6. MULTIPLE ALIGNMENT USING HIDDEN MARKOV MODELS

Step 1. Find $\delta(t, S_i)$ inductively. It is based on formula

$$\delta(t, S_i) = \max_{S_j} \delta(t - 1, S_j)p_{ji}b_i(o_t), \quad 2 \leq t \leq T, 1 \leq j \leq N.$$  

and finally $\max_{Q'} P[Q' \land O] = \max_{S_i} \delta(T, S_i)$. 

Step 2. We recover a state sequence $Q : q_1, q_2, \cdots, q_T$, as follows.

$$q_T = \arg\max_{S_i} \delta(T, S_i).$$

The remaining $q_t$ for $t \leq T - 1$ are found recursively by

$$q_t = \arg\max_{S_i} \delta(t, S_i)p_{ik}b_k(o_{t+1}),$$

where $k$ is the index of $q_{t+1}$, i.e., $q_{t+1} = S_k$.

Remark. In Step 2, if the state $q_t$ is not unique, we arbitrarily take one that gives the maximum. In general, there may be many $Q$’s that maximize $P[Q \land O]$. The Viterbi algorithm just finds one of them.

Example (Con’t). Given the observed sequence $O : 2, 1, 2, 1, 2$, we find the state sequence $\arg\max_{Q'} P[Q' \mid O]$ as follows.

<table>
<thead>
<tr>
<th></th>
<th>$t = 1$</th>
<th>$t = 2$</th>
<th>$t = 3$</th>
<th>$t = 4$</th>
<th>$t = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>$\frac{1}{2^2}$</td>
<td>$\frac{3 \cdot 7}{2^3 \cdot 10}$</td>
<td>$\frac{3^2 \cdot 7}{2^5 \cdot 10^2}$</td>
<td>$\frac{3^3 \cdot 7}{2^6 \cdot 10^3}$</td>
<td>$\frac{3^4 \cdot 7}{2^7 \cdot 10^4}$</td>
</tr>
<tr>
<td>$S_2$</td>
<td>$\frac{3}{2^3}$</td>
<td>$\frac{3^2}{2^5 \cdot 10}$</td>
<td>$\frac{3^2 \cdot 7}{2^6 \cdot 10^2}$</td>
<td>$\frac{3^3 \cdot 7}{2^7 \cdot 10^3}$</td>
<td>$\frac{3^6 \cdot 7}{2^8 \cdot 10^4}$</td>
</tr>
</tbody>
</table>

The desired state sequence is $Q : S_2, S_1, S_1, S_1, S_1$. 
3.6.4 The estimation algorithm

Suppose we are given a set of observed data from an HMM with known topology. We try to estimate the parameters $\lambda$ in that HMM so that $P[\mathcal{O}|\lambda]$ reaches the maximum. Since the parameter space is too large to allow exact computing these parameters, we only have heuristic approaches for the purpose.

Remark. We should assume that the data are generated by some random process that we try to fit with an HMM. Sometimes, it might be possible to achieve a tight fit with an HMM and sometimes it might not.

The Baum-Welch Algorithm

**Input:** Observed sequences $\{\mathcal{O}_d : o_{d1}, o_{d2}, \ldots o_{dT}\}_{d=1}^M$ on alphabet $A$, state set $\{S_j\}_{j=1}^N$.

**Output:** Initial state distribution $\pi = (\pi_j)$ and state transition matrix $P = (p_{ij})$ and emission matrix $B = (b_i(a_j))$.

Initially, parameters $\lambda = (\pi, P, B)$ are chosen from some uniform distribution.

Do loop \{ 
\begin{align*}
\bar{\pi}_i & := \text{the expected proportion of times in state } S_i \text{ at } t = 1, \text{ given } \{\mathcal{O}_d\}_{d=1}^M, 1 \leq i \leq N; \\
\bar{p}_{ij} & := \frac{E(N_{ij}||\mathcal{O}_d)_{d=1}^M)}{E(N_i||\mathcal{O}_d)_{d=1}^M)}, 1 \leq i, j \leq N; \\
\bar{b}_i(a_j) & := \frac{E(N_i(a_j)||\mathcal{O}_d)_{d=1}^M)}{E(N_i||\mathcal{O}_d)_{d=1}^M)}, 1 \leq i \leq N, 1 \leq j \leq M; \\
\bar{\lambda} & := ((\bar{\pi}_i), (\bar{p}_{ij}), (\bar{b}_i(a_j))) \\
\text{if } P[\mathcal{O}_d)_{d=1}^M|\bar{\lambda}] - P[\mathcal{O}_d)_{d=1}^M|\lambda] \leq 10^{-3}, \text{ output } \lambda \text{ and exit; } \\
\lambda & := \bar{\lambda}.
\end{align*}
\}

$N_{ij}$ is the number of times $q_{dt} = S_i$ and $q_{d(t+1)} = S_j$ for some $d$ and $t$; $N_i$ is the number of times $q_{dt} = S_i$ for some $d$ and $t$; $N_i(a)$ is the number of times $q_{dt} = S_i$ and it emits $a$ for some $d$ and $t$. 
3.6. MULTIPLE ALIGNMENT USING HIDDEN MARKOV MODELS

How to compute $N_{ij}, N_i, N_i(a)$'s?

Define

$$\xi_t^{(d)}(i, j) = P[q_{dt} = S_i, q_{d(t+1)} = S_j | \mathcal{O}_d] = \frac{P[q_{dt} = S_i, q_{d(t+1)} = S_j \wedge \mathcal{O}_d]}{P[\mathcal{O}_d]} 1 \leq i, j \leq N$$

$P[\mathcal{O}_d]$ can be calculated efficiently using the forward algorithm. The joint probability in the numerator is calculated using the forwards and backwards variables as follows.

$$P[q_{dt} = S_i, q_{d(t+1)} = S_j \wedge \mathcal{O}_d] = \alpha(t, S_i)p_{ij}b_j(o_{d(t+1)})\beta(t + 1, S_j).$$

Let $I_t^{(d)}$ be the indicator variable defined by

$$I_t^{(d)}(i) = \begin{cases} 1 & \text{if } q_{dt} = S_i \\ 0 & \text{otherwise} \end{cases}$$

The number of times $S_i$ is visited is then $\sum_{d=1}^M \sum_{t=1}^T I_t^{(d)}(i)$. The expected number of times $S_i$ is visited, given $\{\mathcal{O}_d\}_{d=1}^M$, is then,

$$E(N_i|\{\mathcal{O}_d\}_{d=1}^M) = \sum_d \sum_t E(I_t^{(d)}|\mathcal{O}_d) = \sum_d \sum_t (\sum_{j=1}^N \xi_t^{(d)}(i, j)).$$

Similarly, the expected number of transitions from $S_i$ to $S_j$ given $\{\mathcal{O}_d\}_{d=1}^M$ is

$$E(N_{ij}|\{\mathcal{O}_d\}_{d=1}^M) = \sum_d \sum_t \xi_t^{(d)}(i, j).$$

Finally, define indicator random variables $I_t^{(d)}(i, a)$ by

$$I_t^{(d)}(i, a) = \begin{cases} 1 & \text{if } q_{dt} = S_i \text{ and } o_{dt} = a \\ 0 & \text{otherwise} \end{cases}$$

Then,

$$E(N_i(a)|\{\mathcal{O}_d\}_{d=1}^M) = \sum_d \sum_{t:o_{dt}=a} \sum_{j=1}^N \xi_t^{(d)}(i, j).$$
**Overfitting Problem.** A model with too many free parameters cannot be estimated well from a relatively small data set of training sequences.
3.6.5 Alignment by profile HMMs training

1. Modeling protein families An HMM to model protein families has the following structure

This example has “length” 5. In fact, the length is equal to the average length of a sequence in the protein family.

There are three types of states that are drawn in squares, diamonds and circles and are labeled with

- Match states: $m_0, m_1, m_2, m_3, m_4, m_5$,
- Insert states: $i_0, i_1, i_2, i_3, i_4$,
- Delete states: $d_1, d_2, d_3, d_4$.

The process always starts in $m_0$, the start state. A transition always moves to the right as time goes, eventually ending in state $m_5$, the end state. When $m_5$ is reached, the process ends. A match or delete state is never visited more than once.

The emission alphabet $A$ consists of the 20 amino acids together with a “dummy” symbol “delete” (denoted by $\delta$). Delete states $d_i$’s output $\delta$ with probability 1. Each insert and match state has its own emission distribution over the 20 amino acids, and cannot emit a $\delta$. 
Each choice of parameters models a different protein family. The family can be very tight, meaning all sequences in it are very similar, or can be “loose” also, so that there is little similarity between the sequences.

In a protein family, the similarity is high in some positions and low in others. This can be modeled by allowing some match states to have distributions concentrated on a few amino acids while the others have distributions in which all amino acids are roughly equally likely. Thus, for modeling large family of protein sequences, HMMs allow for efficiency and flexibility.

The transition from a match state to an insert state corresponds to the gap open penalty, and the transition from an insert state to itself corresponds to the gap extension penalty. Thus, an HMM model also allows gap penalty to vary along the sequences.

Such HMMs of protein families are called profile HMMs.

Modeling starts with training (or estimating) the parameters of the model using a set of training sequences chosen from a protein family.

The model’s length is chosen to equal to the average length of a sequence in the training set. The estimation procedure uses the Baum-Welch algorithm with all parameters being initialized with uniform distribution (amino acids are emitted with probability 1/20 and transitions of the same type are given 1/2 equal probabilities).
2. Modeling protein subfamilies

When a protein family contains a large number of members, it is sometimes possible to obtain improved results by dividing these proteins into subfamilies. In order to discover $s$ subfamilies, we make $s$ copies of HMM, one for each subfamilies. In this composite HMM, a new start and end state are added for connecting HMM of subfamilies together. Hence, each such HMM is called a *component* of the (composite) HMM. So the composite HMM for the protein family has the following structure:

Such composite models can be trained using a variant of Baum-Welch algorithm.
3. Modeling protein domains and Pfam database

Sometimes, one does not want to build a model of a family of whole proteins like globins, but instead to build a model of a structural domain that occurs as a subsequence in many different kinds of proteins. So, we expect our model to only match a relatively small subsequence of any given protein sequence, with many other unmatched amino acids appearing before and after this subsequence.

We first train an HMM using a set of many unaligned sequences consisting of a specific domain as before. Then, we create 2 new insert states $i_B$ and $i_E$, and a new dummy START state before $i_B$ and a new dummy END state after $i_E$. Eight new transitions are also added to the model. So the HMM architecture can be shown as follows.

Pfam (http://www.sanger.ac.uk/Pfam) uses HMMs to model protein domains. The domains in Pfam are determined based on expert knowledge, sequence similarity, and other protein family databases. To find the protein domains in a query sequence, we run it past each domain in the database using the forward algorithm. When a portion of the query sequence has probability of having been produced by an HMM above a threshold, the domain corresponding to that HMM is reported.

**Remark** HMMs in Pfam is not trained using Baum-Welch algorithm. Their parameters are set using sequence alignment.
4. Multiple Sequence Alignment

Computing multiple sequence alignments for a family of sequences is divided into two steps:

- The sequences to be aligned are used as the training data, to train the parameters of the model.
- For each sequence, the Viterbi algorithm is then used to determine a path most likely to produce that sequence. These paths can then be used to construct an alignment.

**Example.** Consider the sequences $CAEFTTD$ and $CDAEFPTTH$. Suppose the model has length 10 and their most likely paths through the model are

$$m_0m_1m_2m_3m_4d_5d_6m_7m_8m_9m_{10}$$
$$m_0m_1i_1m_2m_3m_4d_5m_6m_7m_9m_{10}$$

Then the alignment induced is found by aligning positions that were generated by the same match state:

$$m_0 \ m_1 \ m_2 \ m_3 \ m_4 \ d_5 \ d_6 \ m_7 \ m_8 \ m_9 \ m_{10}$$
$$C \ \ A \ \ E \ \ F \ \ T \ \ T \ \ D$$

$$m_0 \ i_1 \ m_2 \ m_3 \ m_4 \ d_5 \ m_6 \ m_7 \ m_8 \ m_9 \ m_{10}$$
$$C \ \ D \ \ A \ \ E \ \ F \ \ P \ \ T \ \ T \ \ H$$

so this leads to the alignment

$$C \ - \ A \ E \ F \ - \ T \ T \ D$$
$$C \ D \ A \ E \ F \ P \ T \ T \ H$$
Example (Con’t). Consider the five sequences

\[ \begin{align*}
CAEFTPAVH, \\
CKETTPADH, \\
CAETPDDH, \\
CAEFDDH, \\
CDAEFPDDH,
\end{align*} \]

If their corresponding paths returned by the Viterbi algorithm are

\[ \begin{align*}
m_0m_1m_2m_3m_4m_5m_6m_7m_8m_9m_{10}, \\
m_0m_1m_2m_3m_4m_5m_6m_7m_8m_9m_{10}, \\
m_0m_1m_2m_3m_4d_5m_6m_7m_8m_9m_{10}, \\
m_0m_1m_2m_3m_4d_5d_6m_7m_8m_9m_{10}, \\
m_0m_1i_1m_2m_3m_4d_5m_6m_7m_8m_9m_{10},
\end{align*} \]

then, the induced alignment is

\[ \begin{align*}
C & - A E F T P A V H \\
C & - K E T T P A D H \\
C & - A E - T P D D H \\
C & - A E F - - D D H \\
C & D A E F - - P D D H
\end{align*} \]

Remark. This alignment method gives ambiguous results in some cases. For example, if the model has length two and the sequences are \(ABAC\) and \(ABBAC\) have

\[m_0m_1i_1i_1m_2m_3\]

and \(m_0m_1i_1i_1i_1m_2m_3\)
as paths, then the leading \(A\)'s and trailing \(C\)'s will be aligned, but it is not clear how to align the middle subsequences.