CONTEXT-SENSITIVE HIDDEN MARKOV MODELS FOR MODELING LONG-RANGE DEPENDENCIES IN SYMBOL SEQUENCES

Byung-Jun Yoon, *Student Member*, *IEEE*, and P. P. Vaidyanathan*, *Fellow*, *IEEE*

January 8, 2006

Affiliation: Both authors are with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA, USA.

Contact Information:

Byung-Jun Yoon (Address) Caltech MC 136-93, Pasadena, CA 91125, USA.
(Phone) +1-626-395-2210 (Fax) +1-626-795-8649 (E-mail) bjyoon@systems.caltech.edu

P. P. Vaidyanathan: (Address) Caltech MC 136-93, Pasadena, CA 91125, USA.
(Phone) +1-626-395-4681 (Fax) +1-626-795-8649 (E-mail) ppvnath@systems.caltech.edu

Keywords: Context-sensitive HMM (csHMM), HMM with memory, SCFG, long-range correlation.

EDICS: SSP-SNMD, SSP-SYSM, SSP-APPL

ABSTRACT

The hidden Markov model (HMM) has been widely used in signal processing and digital communication applications. It is well-known for its efficiency in modeling short-term dependencies between adjacent symbols. However, it cannot be used for modeling long-range interactions between symbols that are distant from each other. In this paper, we introduce the concept of context-sensitive HMM. The proposed model is capable of modeling strong pairwise correlations between distant symbols. Based on this model, we propose dynamic programming algorithms that can be used for finding the optimal state sequence and for computing the probability of an observed symbol string. Furthermore, we also introduce a parameter re-estimation algorithm, which can be used for optimizing the model parameters based on the given training sequences.[1]

[1]Work supported in parts by the NSF grant CCF-0428326 and the Microsoft Research Graduate Fellowship.

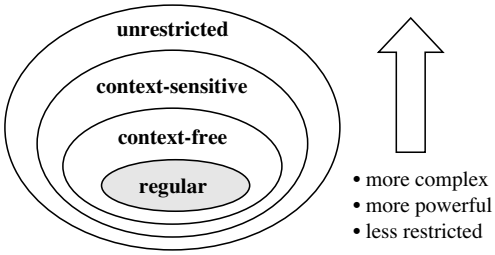

Figure 1: The Chomsky hierarchy of transformational grammars nested according to the restrictions on their production rules.

1 Introduction

Hidden Markov models (HMMs) have been widely used in many fields. They are well-known for their efficiency in modeling short-term dependencies between adjacent symbols, which made them popular in diverse areas. Traditionally, HMMs have been successfully applied to speech recognition, and many speech recognition systems are built upon HMMs and their variants [1, 2]. They have been also widely used in digital communications, and more recently, HMMs have become very popular in computational biology as well. They have been proved to be useful in various problems such as gene identification [3, 4, 5], multiple sequence alignment [5, 6], and so forth. Due to its effectiveness in modeling symbol sequences, the HMM gave rise to a number of useful variants that extend and generalize the basic model [7]-[13].

Although HMMs have a number of advantages, the basic HMM [1, 2] and its variants in [7]-[13] have also inherent limitations. For example, they are capable of modeling sequences with strong correlations between adjacent symbols, but they cannot grasp long-range interactions between symbols that are distant from each other. Therefore, the resulting model always displays sequential dependencies[2], and more complex sequences with non-sequential dependencies cannot be effectively represented using these HMMs.

In his work on transformational grammars, Chomsky categorized all grammars into four classes [14]. These include *regular grammars*, *context-free grammars* (CFG), *context-sensitive grammars* (CSG), and *unrestricted grammars*, in the order of decreasing restrictions on the production rules. The aforementioned classes comprise the so-called *Chomsky hierarchy of transformational grammars* as shown in Fig. 1. The regular grammars are the simplest among the four, and they have the most restricted production rules. HMMs can be viewed as stochastic regular grammars (SRG), according to this hierarchy. Due to the restrictions on their production rules, regular grammars have efficient algorithms such as the Viterbi algorithm [15] for finding the optimal state sequence (popularly used in digital communication receivers), the forward algorithm [1, 2] for computing the probability of an observed symbol string, and the Baum-Welch algorithm [16] for re-estimation of the model parameters. Other transformational grammars that belong to a higher order class in the hierarchy have less restrictions on their production rules, and therefore they have greater descriptive power to represent more complex dependencies between symbols. However, the computational complexity for parsing an observation sequence increases very quickly, which makes the use of higher order grammars sometimes impractical.

[2]By *sequential* dependencies, we imply that the probability that a symbol appears at a certain location depends only on its immediate preceding neighbors.

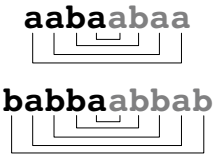

Figure 2: Examples of sequences that are included in the palindrome language.

One interesting language that cannot be represented using regular grammars - or equivalently, using HMMs - is the *palindrome language* [14]. The palindrome language is a language that contains all strings that read the same forwards and backwards. For example, if we consider a palindrome language that uses an alphabet of two letters $\{a, b\}$ for *terminal symbols*[3], it contains all symbol sequences of the form aa, bb, $abba$, $aabbaa$, $abaaba$, and so on. Fig. 2 shows examples of symbol strings that are included in this language. The lines in Fig. 2 that connect two symbols indicate the pairwise correlation between symbols that are distant from each other. This kind of long-range interactions between symbols cannot be described using regular grammars. It is of course possible that a regular grammar generates such palindromes as part of its language. However, we cannot force the model to generate *only such palindromes*. Therefore regular grammars are not able to effectively discriminate palindromic sequences from non-palindromic ones. In fact, in order to describe a palindrome language, we have to use higher-order grammars such as the context-free grammars. Context-free grammars are capable of modeling nested dependencies between symbols that are shown in Fig. 2.

In this paper, we introduce the idea of context-sensitive hidden Markov model (csHMM), which is an extension of the traditional HMM. The csHMM is capable of modeling long-range correlations, by rendering certain states in the model context-sensitive. The proposed model has several advantages over the existing models including the stochastic context-free grammars (SCFG), which will be demonstrated later. The organization of this paper is as follows. In Sec. 2, we elaborate on the basic concept of context-sensitive HMMs. It will be explained how they can represent complex dependencies between symbols that are far away from each other. In Sec. 3, we propose a dynamic programming algorithm that can be used for finding the optimal state sequence of an observed symbol string, based on the given model. In Sec. 4, the scoring algorithm for csHMM is introduced, which can compute the probability of an observed sequence in an efficient way. Moreover, we also introduce the outside-algorithm that can be used along with the scoring algorithm for training the model based on the given sequences. The parameter re-estimation algorithm that is used for training csHMMs is proposed in Sec. 5. Sec. 6 provides an example that illustrates the effectiveness of the proposed algorithms. In Sec. 7, we discuss several interesting issues regarding the descriptive power of the csHMM. We also compare the proposed model with other variants of the traditional HMM and other stochastic grammars. The paper is concluded in Sec. 8.

It has to be noted that the *context-sensitive HMMs* proposed in this paper are not related to the so-called *context-dependent HMMs* that have been widely used in speech recognition [17, 18, 19]. They are regular HMMs, whose basic building blocks are built by considering the phonetic context, hence called context-dependent HMMs.

[3] A transformational grammar has two kinds of symbols, namely, *non-terminal symbols* and *terminal symbols*. Non-terminal symbols are similar to the states in HMMs that are hidden to us, and terminal symbols are the symbols that we observe.

2 Context-Sensitive Hidden Markov Models

The context-sensitive HMM can be viewed as an extension of the traditional HMM, where some of the states are equipped with auxiliary memory [20, 21]. Symbols that are emitted at certain states are stored in the memory, and the stored data serves as the *context* which affects the emission probabilities and the transition probabilities of the model. This context-sensitive property increases the descriptive power of the model significantly, compared to the traditional HMM. Let us first formally define the basic elements of the context-sensitive HMM.

2.1 Basic elements of a csHMM

Similar to the traditional HMMs, the csHMM is also a *doubly-stochastic process*, which consists of a non-observable process of hidden states and a process of observable symbols. The process of the hidden states is governed by state-transition probabilities that are associated with the model, and the observation process is linked to the hidden process via emission probabilities of the observed symbol that is conditioned on the hidden state. A csHMM can be characterized by the following elements.

2.1.1 Hidden states

We assume that the csHMM has M distinct states. The set of hidden states $\mathcal{V}$ is defined as

$$\mathcal{V} = \mathcal{S} \cup \mathcal{P} \cup \mathcal{C} \cup \{\text{start},\ \text{end}\}, \tag{1}$$

where {start, end} denote the set of special states that are used to denote the *start* state and the *end* state of the model. As can be seen in (1), there are three different classes of states, namely, *single-emission states* S_n, *pairwise-emission states* P_n, and *context-sensitive states* C_n. $\mathcal{S}$ is the set of single-emission states

$$\mathcal{S} = \{S_1, S_2, \ldots, S_{M_2}\}, \tag{2}$$

where M_2 is the number of single-emission states in the model. Similarly, $\mathcal{P}$ and $\mathcal{C}$ denote the set of pairwise-emission states and the set of context-sensitive states

$$\mathcal{P} = \{P_1, P_2, \ldots, P_{M_1}\}, \quad \mathcal{C} = \{C_1, C_2, \ldots, C_{M_1}\}. \tag{3}$$

As shown in (3), the number of pairwise-emission states is the same as the number of context-sensitive states. Therefore, we have $M = 2M_1 + M_2 + 2$ hidden states in total. The states P_n and C_n always exist in pairs. For example, if there are two pairwise-emission states P_1 and P_2 in the model, then the HMM is required to have also two context-sensitive states C_1 and C_2. The two states P_n and C_n are associated with a separate memory element Z_n, such as a stack or a queue. We may also use other memory types depending on the application. Fig. 3 shows an example where P_n and C_n are associated with a stack Z_n. We use the same notation Z_n for both the memory and the data stored in the memory (the *context*), for simplicity.

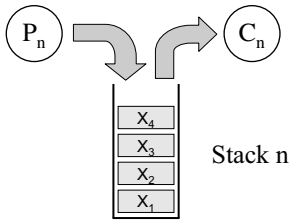

Figure 3: The states P_n and C_n associated with a stack Z_n.

Differences between the three classes of states. The differences between the three classes of states are as follows.

Single-emission state $\mathbf{S_n}$. The single-emission state S_n is identical to the regular hidden state in the traditional HMMs. As we enter the state, it emits an observable symbol according to the associated emission probabilities. After the emission, S_n makes a transition to the next state according to the specified transition probabilities.

Pairwise-emission state $\mathbf{P_n}$. The pairwise-emission state P_n is almost identical to the single-emission state S_n, except that the symbols emitted at P_n are stored in the auxiliary memory Z_n dedicated to P_n and C_n. The data stored in the memory affects the emission probabilities and the transition probabilities of C_n in the future. After storing the emitted symbol in the memory, a transition is made to the next state according to the transition probabilities of P_n.

Context-sensitive state $\mathbf{C_n}$. The context-sensitive state C_n is considerably different from the other states, in the sense that its emission probabilities and the transition probabilities are not fixed. In fact, these probabilities depend on the *context*, or the data stored in the associated memory Z_n, which is the reason why C_n is called a context-sensitive state. When entering C_n, it first accesses the memory Z_n and retrieves a symbol x. Once the symbol is retrieved, the emission probabilities of C_n are adjusted according to the value of x. For example, we may adjust the emission probabilities of C_n such that it emits the same symbol x with high probability (possibly, with probability one). Transition probabilities at C_n also depend on the context, which will be explained later.

We denote the hidden state process as $\mathbf{s} = s_1 s_2 \dots s_L$, where s_i is the state at time i and L is the length of the entire sequence. Each state takes a value from $s_i \in \mathcal{V} - \{\text{start}, \text{end}\}$. The virtual start state s_0 and the end state s_{L+1} are assumed to be $s_0 = \textit{start}$ and $s_{L+1} = \textit{end}$.

2.1.2 Observation symbols

We denote the observation process as $\mathbf{x} = x_1 x_2 \dots x_L$, where x_i is the observed symbol at time i. Each symbol x_i takes a value from an alphabet $x_i \in \mathcal{A}$. Note that the virtual start state s_0 and the end state s_{L+1} do not make any emission.

2.1.3 Transition probabilities

Let us define the probability that the model will make a transition from a state $s_i = v$ to the next state $s_{i+1} = w$. For $v \in \mathcal{S} \cup \mathcal{P}$, we define the probability as

$$P(s_{i+1} = w | s_i = v) = t(v, w). \tag{4}$$

Note that the transition probabilities are stationary and do not depend on the time index i. As mentioned earlier, the transition probabilities at a context-sensitive state C_n depend on the context Z_n. C_n uses two different sets of transition probabilities, depending on whether the associated memory Z_n is empty or not. For each context-sensitive state C_n, we define the following sets

$$\begin{aligned} \mathcal{E}_n &= \{ \text{ Subset of } \mathcal{V} \text{ that contains the states to which } C_n \text{ can make} \\ &\qquad \text{transitions when the associated memory element } Z_n \text{ is empty } \} \end{aligned} \tag{5}$$

$$\begin{aligned} \mathcal{F}_n &= \{ \text{ Subset of } \mathcal{V} \text{ that contains the states to which } C_n \text{ can make} \\ &\qquad \text{transitions when the associated memory element } Z_n \text{ is not empty } \}, \end{aligned} \tag{6}$$

where $\mathcal{E}_n \cap \mathcal{F}_n = \varnothing$. At a context-sensitive state C_n, the memory is examined after making the emission. If the memory is empty, $v = C_n$ can make a transition only to $w \in \mathcal{E}_n$. Similarly, if the memory is not empty, $v = C_n$ makes a transition to $w \in \mathcal{F}_n$. Based on this setting, we define the two sets of transition probabilities when $v \in \mathcal{C}$ as follows

$$P(s_{i+1} = w | s_i = v, Z_n) = \begin{cases} t_e(v, w) & \text{if } Z_n \text{ is empty} \\ t_f(v, w) & \text{if } Z_n \text{ is not empty.} \end{cases} \tag{7}$$

Since $\mathcal{E}_n \cap \mathcal{F}_n = \varnothing$, the probabilities $t_e(v, w)$ and $t_f(v, w)$ cannot have non-zero values at the same time. Therefore, we can let $t(v, w) = t_e(v, w) + t_f(v, w)$ without any ambiguity. Now, the transition probability from $s_i = v \in \mathcal{C}$ to $s_{i+1} = w$ can be simplified as

$$P(s_{i+1} = w | s_i = v, Z_n) = t(v, w). \tag{8}$$

Note that we have $\sum_{w \in \mathcal{E}_n} t(v, w) = 1$ and $\sum_{w \in \mathcal{F}_n} t(v, w) = 1$ in this case. The probability $t(\mathit{start}, v)$ is used to define the initial state distribution $P(s_1 = v)$, and $t(w, \mathit{end})$ denotes the probability that the HMM will terminate after the state w.

Preventing degeneracies. The restrictions on the states to which a context-sensitive state $v \in \mathcal{C}$ is allowed to make transitions depending on the context, can be conveniently used to maintain the number of P_n and that of C_n identical in a state sequence. In this way, we can prevent degenerate situations due to a mismatch of the two states. Let $\mathbf{s} = s_1 s_2 \ldots s_L$ be a feasible state sequence of an observed symbol string $\mathbf{x} = x_1 x_2 \ldots x_L$. The csHMM should be constructed such that the number of occurrences of P_n in the sequence $\mathbf{s}$ is kept the same as the number of occurrences of C_n in $\mathbf{s}$. This restriction is reasonable for the following reasons. In the first place, if there are more C_n states than there are P_n states, the emission probabilities of the context-sensitive state C_n cannot be properly determined. On the other hand, if there are more P_n states than C_n states, the symbols that were emitted at the "surplus" P_n states do not affect the probabilities in the model at all, hence they may be simply replaced by single-emission states.

2.1.4 Emission probabilities

The probability of observing a symbol $x_i = x$ depends on the underlying hidden state $s_i = v$. For $v \in \mathcal{S} \cup \mathcal{P}$, this emission probability can be defined as

$$P(x_i = x | s_i = v) = e(x|v). \tag{9}$$

For $v \in \mathcal{C}$, the emission probability depends on both $s_i = v$ and the context Z_n, hence it is defined as

$$P(x_i = x | s_i = v, Z_n) = e(x|v, Z_n). \tag{10}$$

In case the emission probability depends only on a single symbol x_p in the memory Z_n (e.g. if Z_n uses a stack, x_p may be the symbol on the top of the stack), the emission probability in (10) can be simply written as $e(x|v, x_p)$.

2.2 Constructing a csHMM

By using the proposed context-sensitive HMM, we can easily construct a simple model that generates *only* palindromes. For example, we may use the structure shown in Fig. 4. As can bee seen in Fig. 4, there are three hidden states S_1, P_1, and C_1 in the model, where the state-pair (P_1, C_1) is associated with a stack. Initially, the model begins at the pairwise-emission state P_1. It makes several self-transitions to generate a number of symbols, which are pushed onto the stack. At some point, it makes a transition to the context-sensitive state C_1. Once we enter the context-sensitive state C_1, the emission probabilities and the transition probabilities of C_1 are adjusted, such that the state always emits the symbol on the top of the stack and makes self-transitions until the stack becomes empty. In this way, C_1 emits the same symbols as were emitted by P_1, but in the reverse order, since the stack is a last-in-first-out (LIFO) system. If we denote the number of symbols that were emitted by P_1 as N, the generated string will always be a palindrome of the form $x_1 \dots x_N x_N \dots x_1$ (even length sequence) or $x_1 \dots x_N x_{N+1} x_N \dots x_1$ (odd length sequence).

In the following discussions, we mainly focus on those context-sensitive HMMs that generate sequences with nested interactions. These models include the ones that generate palindromic sequences as illustrated in Fig. 4. As in Fig. 4, we assume that every state-pair (P_n, C_n) is associated with a stack. Based on these csHMMs, we describe efficient algorithms that can be used for sequence analysis.

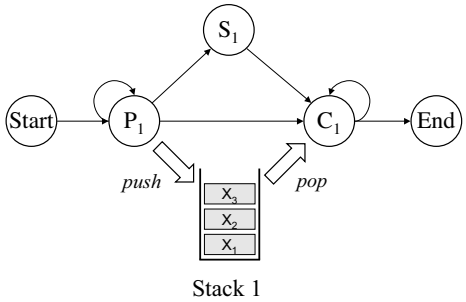

Figure 4: An example of a context-sensitive HMM that generates only palindromes.

3 Finding the Most Probable Path

Let us consider an observation sequence $\mathbf{x} = x_1 x_2 \ldots x_L$. As described in Sec. 2, we denote the underlying state of x_i as s_i. Assuming that there are M distinct states in the model, we have M^L different paths. Given the observation sequence $\mathbf{x}$, how can we find the path that is most probable among the M^L distinct paths? This problem is tradictionally called the *optimal alignment problem*, since we are trying to find the best alignment between the observed symbol string and the given HMM.

One way to find the most probable path would be to compute the probabilities of all paths, and pick the one with the highest probability. However, this approach is impractical, since the number of paths increases exponentially with the length L of the sequence. When using traditional HMMs, this problem can be solved very efficiently by the Viterbi algorithm [15], which is widely used in digital communication receivers. The Viterbi algorithm exploits the fact that if $s_1 \ldots s_{i-1} s_i$ is the optimal path for $x_1 \ldots x_{i-1} x_i$ among all paths that end with the state s_i, then $s_1 \ldots s_{i-1}$ must be the optimal path for $x_1 \ldots x_{i-1}$ among all paths that end with the state s_{i-1}. Therefore, in order to find the optimal path for $x_1 \ldots x_i$ with $s_i = v$, we only have to consider the M optimal paths for $x_1 \ldots x_{i-1}$ that end with $s_{i-1} = 1, \ldots, M$, the transition probability from each of these states to the state $s_i = v$, and the probability of emitting the symbol x_i at the state s_i. This makes the computational complexity of the Viterbi algorithm only $O(LM^2)$, which is considerably better than $O(LM^L)$ of the exhaustive search.

Unfortunately, the same intuition does not hold for context-sensitive HMMs. Since the emission probabilities and the transition probabilities of context-sensitive states C_n depend on the previously emitted symbols at the pairwise-emission states P_n, we have to keep track of the previous states in order to compute the probability of a certain path. Therefore, the optimal path for $x_1 \ldots x_i$ cannot be found simply by considering the optimal paths for $x_1 \ldots x_{i-1}$ and extending it.

In order to see this, let us consider the example in Fig. 5. This context-sensitive HMM has three hidden states P_1, C_1, and S_1, where each of these states emits a symbol in the alphabet $\mathcal{A} = \{a, b\}$. The emission probabilities and the transition probabilities of P_1 and S_1 are shown in the figure. The symbols emitted at P_1 are pushed onto the stack, and this data affects the probabilities at the state C_1. Once we enter the context-sensitive state C_1, a symbol is popped out from the stack and is emitted. After the emission, the stack is examined to check whether it is empty. If it is empty, the model terminates. Otherwise, the model makes a transition back to C_1 and continues emitting the symbols that are stored in the stack. Now, let us consider the symbol sequence *abbba*. Assuming that this string comes from the model in Fig. 5, what is the

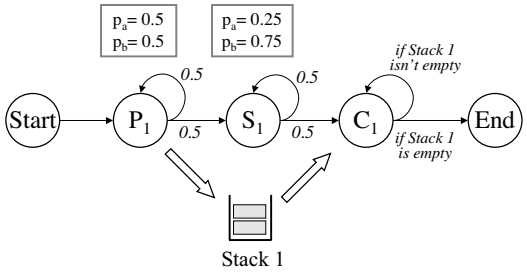

Figure 5: An example of a simple context-sensitive HMM.

most probable path $\mathbf{s}^*$? It is not difficult to see that there are only two feasible paths: $\mathbf{s}_1 = P_1S_1S_1S_1C_1$ and $\mathbf{s}_2 = P_1P_1S_1C_1C_1$. Since both paths pass the state S_1 in the middle, let us first consider the optimal path for the first three symbols *abb*. We denote the sub-paths of $\mathbf{s}_1$ and $\mathbf{s}_2$ up to the third symbol as $\hat{\mathbf{s}}_1 = P_1S_1S_1$ and $\hat{\mathbf{s}}_2 = P_1P_1S_1$, respectively. If we compute the probabilities of $\hat{\mathbf{s}}_1$ and $\hat{\mathbf{s}}_2$, we get

$$P(\hat{\mathbf{s}}_1) = \frac{1}{2} \times \frac{1}{2} \times \frac{3}{4} \times \frac{1}{2} \times \frac{3}{4} = \frac{9}{128} \tag{11}$$

and

$$P(\hat{\mathbf{s}}_2) = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{3}{4} = \frac{6}{128}, \tag{12}$$

hence the optimal path for the first three symbols *abb* is $\hat{\mathbf{s}}_1$. However, if we compute the probabilities of the two paths $\mathbf{s}_1$ and $\mathbf{s}_2$, we obtain

$$P(\mathbf{s}_1) = \frac{1}{2} \times \frac{1}{2} \times \frac{3}{4} \times \frac{1}{2} \times \frac{3}{4} \times \frac{1}{2} \times \frac{3}{4} \times \frac{1}{2} \times 1 \times 1 = \frac{27}{2048} \tag{13}$$

and

$$P(\mathbf{s}_2) = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{3}{4} \times \frac{1}{2} \times 1 \times 1 \times 1 \times 1 \ = \frac{48}{2048}, \tag{14}$$

which shows that the optimal path for *abbba* is $\mathbf{s}_2$. Apparently, the globally optimal path $\mathbf{s}^* = \mathbf{s}_2$ is not an extension of $\hat{\mathbf{s}}_1$, and this example clearly demonstrates that the Viterbi algorithm cannot be used for finding the most probable path in context-sensitive HMMs.

3.1 Alignment of csHMM

Although the Viterbi algorithm cannot be used for finding the optimal path in a context-sensitive HMM, we can develop a polynomial-time algorithm that solves the alignment problem in a recursive manner, similar to the Viterbi algorithm. The proposed algorithm is conceptually similar to the Cocke-Younger-Kasami (CYK) algorithm [22, 23] that can be used for parsing SCFGs. The main reason why the Viterbi algorithm cannot be used in context-sensitive HMMs is because the interactions between symbols are not sequential. Since the Viterbi algorithm basically considers only sequential depdendencies, it cannot take care of nested interactions between distant symbols. However, if we implement an algorithm that starts from the inside of the given sequence and proceeds to the outward direction by taking the nested interactions into account, it is possible to find the optimal state sequence in a recursive manner.

When searching for the most probable state sequence, we assume that all pairwise interactions between P_n and C_n are nested and they do not cross each other, as mentioned earlier. Fig. 6 illustrates several examples of interactions that are allowed as well as those that are prohibited. The nodes in the figure denote the observed symbols in the sequence, and the dotted lines that connect two symbols indicate the pairwise interactions between them. The sequence in Fig. 6 (a)$\sim$(c) shows sequences with nested dependencies. On the other hand, the example in Fig. 6 (d) shows a sequence with a crossing interaction, which is not considered in this case.

Before describing the algorithm, let us first define the variables that are needed in the proposed algorithm. $\mathbf{x} = x_1 \dots x_L$ is the observation sequence and $\mathbf{s} = s_1 \dots s_L$ is the underlying state sequence. We assume that the csHMM has M distinct states, which we simply denote by $\mathcal{V} = \{1, 2, \dots, M\}$. The state

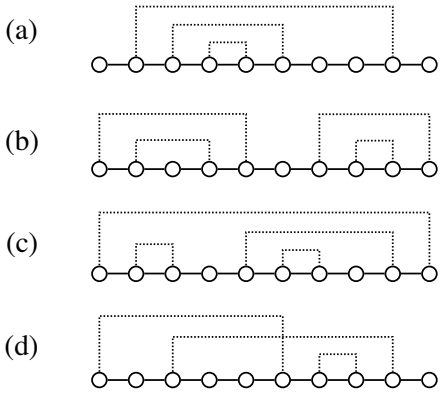

Figure 6: Examples of interactions in a symbol string. The dotted lines indicate the pairwise dependencies between symbols. (a), (b), (c) Nested interactions. (d) Crossing interactions.

$v = 1$ denotes the *start* state of the HMM and $v = M$ denotes the *end* state. For $v \in \mathcal{P} \cup \mathcal{C}$, we define $\bar{v}$ as the complementary state of v as follows,

$$v = P_n \rightarrow \bar{v} = C_n, \qquad v = C_n \rightarrow \bar{v} = P_n. \tag{15}$$

The emission probability of a symbol x at a state v is defined as $e(x|v)$ for $v \in \mathcal{S} \cup \mathcal{P}$, and $e(x|v, x_p)$ for $v \in \mathcal{C}$, where x_p is the symbol that was previously emitted at the corresponding pairwise-emission state $\bar{v}$. The transition probability from v to w is defined as $t(v, w)$. Finally, let us define $\gamma(i, j, v, w)$ to be the log-probability of the optimal path among all sub-paths $s_i \dots s_j$ with $s_i = v$ and $s_j = w$. In computing $\gamma(i, j, v, w)$, we consider only those paths where all the pairwise-emission states P_n in the $s_i \dots s_j$ are paired with the corresponding context-sensitive states C_n. Examples of sub-paths that are considered in computing $\gamma(i, j, v, w)$ are shown in Fig. 7 (a). The paths shown in Fig. 7 (b) are not considered due to unpaired P_n or C_n states, or due to crossing interactions. The variable $\gamma(i, j, v, w)$ will ultimately lead to the probability $\log P(\mathbf{x}, \mathbf{s}^*|\Theta)$, where $\mathbf{s}^*$ is the optimal path that satisfies

$$\mathbf{s}^* = \arg\max_{\hat{\mathbf{s}}} P(\mathbf{x}, \mathbf{s} = \hat{\mathbf{s}}|\Theta), \tag{16}$$

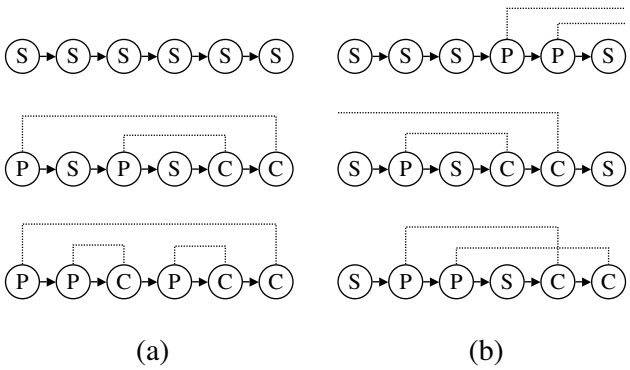

Figure 7: Examples of state sequences (a) that are considered in computing $\gamma(i, j, v, w)$ and (b) those that are not considered.

where Θ is the set of model parameters. Additionally, we define the variables $\lambda_\ell(i,j,v,w)$ and $\lambda_r(i,j,v,w)$ that will be used for tracing back the optimal path $\mathbf{s}^*$.

3.1.1 Computing the Probability of the Optimal Path

Now, the alignment algorithm can be described as follows.

1) Initialization

For $i = 1, \ldots, L, v = 2, \ldots, M-1$.

$$\begin{aligned}
\gamma(i,i,v,v) &= \begin{cases} \log e(x_i|v) & v \in \mathcal{S} \\ -\infty & \text{otherwise} \end{cases} \\
\lambda_\ell(i,i,v,v) &= (0,0,0,0) \\
\lambda_r(i,i,v,v) &= (0,0,0,0)
\end{aligned}$$

2) Iteration

For $i = 1, \ldots, L-1, j = i+1, \ldots, L$ and $v = 2, \ldots, M-1, w = 2, \ldots, M-1$.

(i) $v \in \mathcal{C}$ or $w \in \mathcal{P}$

$$\begin{aligned}
\gamma(i,j,v,w) &= -\infty \\
\lambda_\ell(i,j,v,w) &= (0,0,0,0) \\
\lambda_r(i,j,v,w) &= (0,0,0,0)
\end{aligned}$$

(ii) $v \in \mathcal{P}, w \in \mathcal{S}$

$$\begin{aligned}
\gamma(i,j,v,w) &= \max_u \Big[\gamma(i,j-1,v,u) + \log t(u,w) + \log e(x_j|w)\Big] \\
u^* &= \arg\max_u \Big[\gamma(i,j-1,v,u) + \log t(u,w) + \log e(x_j|w)\Big] \\
\lambda_\ell(i,j,v,w) &= (i,j-1,v,u^*) \\
\lambda_r(i,j,v,w) &= (j,j,w,w)
\end{aligned}$$

(iii) $v \in \mathcal{S}, w \in \mathcal{C}$

$$\begin{aligned}
\gamma(i,j,v,w) &= \max_u \Big[\log e(x_i|v) + \log t(v,u) + \gamma(i+1,j,u,w)\Big] \\
u^* &= \arg\max_u \Big[\log e(x_i|v) + \log t(v,u) + \gamma(i+1,j,u,w)\Big] \\
\lambda_\ell(i,j,v,w) &= (i,i,v,v) \\
\lambda_r(i,j,v,w) &= (i+1,j,u^*,w)
\end{aligned}$$

(iv) $v = P_n, w = C_m$ $(n \neq m), j < i+3$

$$\begin{aligned}
\gamma(i,j,v,w) &= -\infty \\
\lambda_\ell(i,j,v,w) &= (0,0,0,0) \\
\lambda_r(i,j,v,w) &= (0,0,0,0)
\end{aligned}$$

(v) $v = P_n, w = C_m$ $(n \neq m), j \geq i+3$

$$\begin{aligned}
\gamma(i,j,v,w) &= \max_u \left(\max_{k=i+1,\ldots,j-2} \Big[\gamma(i,k,v,\bar{v}) + \log t(\bar{v},u) + \gamma(k+1,j,u,w) \Big] \right) \\
(k^*,u^*) &= \arg\max_{(u,k),k=i+1,\ldots,j-1} \Big[\gamma(i,k,v,\bar{v}) + \log t(\bar{v},u) + \gamma(k+1,j,u,w) \Big] \\
\lambda_\ell(i,j,v,w) &= (i,k^*,v,\bar{v}) \\
\lambda_r(i,j,v,w) &= (k^*+1,j,u^*,w)
\end{aligned}$$

(vi) $v = P_n, w = C_n, j = i+1$

$$\begin{aligned}
\gamma(i,j,v,w) &= \log e(x_i|v) + \log t(v,w) + \log e(x_j|w,x_i) \\
\lambda_\ell(i,j,v,w) &= (0,0,0,0) \\
\lambda_r(i,j,v,w) &= (0,0,0,0)
\end{aligned}$$

(vii) $v = P_n, w = C_n, j > i+1$

$$\begin{aligned}
\gamma_1 &= \max_u \left(\max_{k=i+1,\ldots,j-2} \Big[\gamma(i,k,v,\bar{v}) + \log t(\bar{v},u) + \gamma(k+1,j,u,w) \Big] \right) \\
(k^*,u^*) &= \arg\max_{(u,k),k=i+1,\ldots,j-1} \Big[\gamma(i,k,v,\bar{v}) + \log t(\bar{v},u) + \gamma(k+1,j,u,w) \Big] \\
\gamma_2 &= \max_{u_1,u_2} \Big[\log e(x_i|v) + \log t(v,u_1) \\
&\qquad + \gamma(i+1,j-1,u_1,u_2) + \log t(u_2,w) + \log e(x_j|w,x_i) \Big] \\
(u_1^*,u_2^*) &= \arg\max_{(u_1,u_2)} \Big[\log e(x_i|v) + \log t(v,u_1) \\
&\qquad + \gamma(i+1,j-1,u_1,u_2) + \log t(u_2,w) + \log e(x_j|w,x_i) \Big] \\
\gamma(i,j,v,w) &= \max(\gamma_1,\gamma_2)
\end{aligned}$$

If $\gamma_1 \geq \gamma_2$,

$$\begin{aligned}
\lambda_\ell(i,j,v,w) &= (i,k^*,v,w) \\
\lambda_r(i,j,v,w) &= (k^*+1,j,u^*,w).
\end{aligned}$$

Otherwise,

$$\begin{aligned}
\lambda_\ell(i,j,v,w) &= (i+1,j-1,u_1^*,u_2^*) \\
\lambda_r(i,j,v,w) &= (0,0,0,0).
\end{aligned}$$

(viii) $v \in \mathcal{S}, w \in \mathcal{S}$

In this case, the variable $\gamma(i,j,v,w)$ can be updated using any of the update formulae in (ii) or (iii).

3) Termination

$$\begin{aligned}
\log P(\mathbf{x},\mathbf{s}^*|\Theta) &= \max_{v,w} \Big[\log t(1,v) + \gamma(1,L,v,w) + \log t(w,M) \Big] \\
(v^*,w^*) &= \arg\max_{(v,w)} \Big[\log t(1,v) + \gamma(1,L,v,w) + \log t(w,M) \Big] \\
\lambda^* &= (1,L,v^*,w^*) \qquad \square
\end{aligned}$$

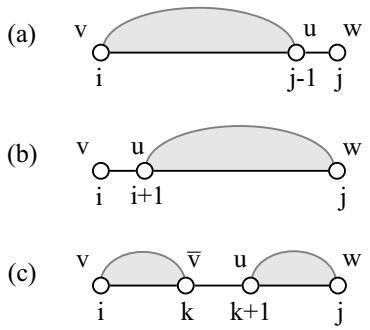

Figure 8: Illustration of the iteration step of the algorithm.

As shown in the **initialization** step of the algorithm, we start by initializing the values of $\gamma(i, i, v, v)$ for $i = 1, 2, \ldots, L$ and $v = 2, 3, \ldots, M-1$. Since we consider only state sequences where all the pairwise-emission states and the context-sensitive states are paired, the value of $\gamma(i, i, v, v)$ is set to $-\infty$ for $v \in \mathcal{P}$ or $v \in \mathcal{C}$. For single-emission states $v \in \mathcal{S}$, $\gamma(i, i, v, v)$ is simply the logarithm of the emission probability of the symbol x_i at state v. Therefore, we set $\gamma(i, i, v, v) = \log e(x_i|v)$ for $v \in \mathcal{S}$.

Now, let us consider the **iteration** step. As we can see in (i) and (iv), the variable $\gamma(i, j, v, w)$ is set to $-\infty$, whenever the states P_n and C_n do not form pairs. For example, in case (i), if the leftmost state s_i of the sub-path $s_i \ldots s_j$ is a context-sensitive state, it cannot be paired with the corresponding pairwise-emission state, since there are no more states to the left of s_i. This is also true when the rightmost state s_j is a pairwise-emission state. In case (iv), the state sequence is either $s_i s_{i+1}$ or $s_i s_{i+1} s_{i+2}$. As $s_i = P_n$ and $s_j = C_m$ where $n \neq m$, the states s_i and s_j cannot form a pair. Moreover, since there are not enough states between s_i and s_j such that both s_i and s_j can form pairs respectively, the probability of such a state sequence is zero. Case (ii) in the **iteration** step deals with the case when $s_i = v$ is a pairwise-emission state while $s_j = w$ is a single-emission state. Since there can be no interaction between s_j and any other state s_k $(i \leq k \leq j-1)$, all the pairwise-emission states and the corresponding context-sensitive states should form pairs inside the sub-path $s_i \ldots s_{j-1}$. As $\gamma(i, j-1, v, u)$ is the log-probability of the optimal path among all feasible paths $s_i \ldots s_{j-1}$, we can compute $\gamma(i, j, v, w)$ by extending $\gamma(i, j-1, v, u)$ to the right by one symbol. We first take the summation of $\gamma(i, j-1, v, u)$ and $\log t(u, w)$ and $\log e(x_i|w)$, and then compute the maximum value of this sum over all u, as described in (ii) of the **iteration** step. Fig. 8 (a) illustrates this case, where the shaded area indicates that all P_n and C_n states are paired inside the sub-path $s_i \ldots s_{j-1}$. Similar reasoning holds also for the case when $s_i = v$ is a single-emission state and $s_j = w$ is a context-sensitive state. In this case, $\gamma(i, j, v, w)$ can be obtained by extending $\gamma(i+1, j, u, w)$ as in (iii) of the **iteration** step. This is illustrated in Fig. 8 (b).

Fig. 8 (c) depicts the case when $s_i = P_n$ and $s_j = C_m$, where $n \neq m$. In this case, the pairwise-emission state s_i and the context-sensitive state s_j cannot form a pair. Therefore $s_i = P_n$ should pair with $s_k = \bar{v} = C_n$ for some k $(i+1 \leq k \leq j-2)$. Similarly, $s_j = C_m$ should form a pair with $s_\ell = \bar{w} = P_m$ for some ℓ $(k+1 \leq \ell \leq j-1)$. Consequently, all pairwise-emission states and context-sensitive states inside $s_i \ldots s_k$ and $s_{k+1} \ldots s_j$ have to exist in pairs. Therefore, we can obtain $\gamma(i, j, v, w)$ by adding $\gamma(i, k, v, \bar{v})$, the transition probability $\log t(\bar{v}, u)$, and $\gamma(k+1, j, u, w)$, and maximizing this sum over all u and k, as

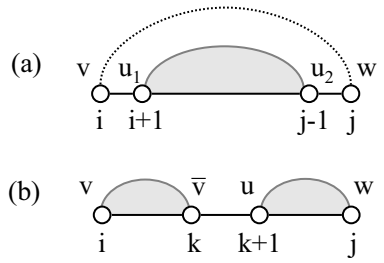

Figure 9: Illustration of the iteration step of the algorithm for the case when $s_i = P_n$ and $s_j = C_n$.

shown in (v).

Finally, let us focus on the case when $s_i = P_n$ and $s_j = C_n$. If $j = i + 1$, we can simply compute $\gamma(i, j, v, w)$ as in (vi) of the **iteration** step. As s_i pairs with s_j, we consider the emission of the symbols x_i and x_j at the same time. In this way, we know the emitted symbol x_i, and therefore the emission probabilities at the context-sensitive state $s_j = C_n$ can be decided correspondingly. When $j \neq i + 1$, the situation is a little bit more complicated. In this case, we have the following two possibilities. One possibility is that s_i forms a pair with s_j as shown in Fig. 9 (a). The dotted line that connects s_i and s_j indicates the pairwise interaction between the two symbols. Since s_i and s_j form a pair, the pairwise-emission states and the context-sensitive states in $s_{i+1} \dots s_{j-1}$ should necessarily exist in pairs. Therefore, the log-probability of the most probable path, where $s_i = P_n$ and $s_j = C_n$ form a pair can be computed as follows

$$\max_{u_1,u_2} \Big[\log e(x_i|v) + \log t(v, u_1) + \gamma(i + 1, j - 1, u_1, u_2) + \log t(u_2, w) + \log e(x_j|w, x_i) \Big]. \tag{17}$$

Another possibility is that $s_i = P_n$ pairs with $s_k = C_n$ for some k between $i + 1$ and $j - 2$. In this case, $s_j = C_n$ has to pair with $s_\ell = P_n$ for some ℓ between $k + 1$ and $j - 1$. Therefore, all P_n and C_n states inside $s_i \dots s_k$ and $s_{k+1} \dots s_j$ have to exists in pairs as illustrated in Fig. 9 (b). The log-probability of all feasible paths, where $s_i = P_n$ does not pair with $s_j = C_n$ can be computed by

$$\max_{u} \left(\max_{k=i+1,\dots,j-2} \Big[\gamma(i, k, v, \bar{v}) + \log t(\bar{v}, u) + \gamma(k + 1, j, u, w) \Big] \right). \tag{18}$$

By comparing (17) and (18) as in (vii) of the **iteration** step, we can compute the log-probability of the most probable path among all sub-paths $s_i \dots s_j$ with $s_i = P_n$ and $s_j = C_n$.

Once we have completed the **iteration** step, the log-probability $\log P(\mathbf{x}, \mathbf{s}^*|\Theta)$ of the most probable path $\mathbf{s}^*$ can be computed by comparing $\gamma(1, L, v, w)$ for all $v, w = 2, 3, \dots, M - 1$. This is shown in the **termination** step.

3.1.2 Trace-Back

Now that we have obtained the log-probability of the optimal path, we can trace-back the path $\mathbf{s}^*$ that gave rise to this probability. The variables $\lambda_\ell(i, j, v, w)$ and $\lambda_r(i, j, v, w)$ are used in the trace-back procedure, and we also need a stack T. For notational convenience, let us define $\lambda_t = (i, j, v, w)$. The procedure can be described as the following.

Case	Complexity for one iteration	Number of iterations	Overall complexity
i	$O(1)$	$O(L^2M_1M)$	$O(L^2M_1M)$
ii	$O(M)$	$O(L^2M_1M_2)$	$O(L^2M_1M_2M)$
iii	$O(M)$	$O(L^2M_1M_2)$	$O(L^2M_1M_2M)$
iv	$O(1)$	$O(LM_1^2)$	$O(LM_1^2)$
v	$O(ML)$	$O(L^2M_1^2)$	$O(L^3M_1^2M)$
vi	$O(1)$	$O(LM_1)$	$O(LM_1)$
vii	$O(ML)+O(M^2)$	$O(L^2M_1)$	$O(L^3M_1M)+O(L^2M_1M^2)$
viii	$O(M)$	$O(L^2M_2^2)$	$O(L^2M_2^2M)$

Table 1: Computational complexity of the csHMM alignment algorithm.

1) Initialization

$s_i = 0$ $(i = 1, 2, \ldots, L)$.

Push λ^* onto T.

2) Iteration

Pop $\lambda_t = (i, j, v, w)$ from stack T.

If $\lambda_t \neq (0, 0, 0, 0)$

If $s_i = 0$ then $s_i = v$.

If $s_j = 0$ then $s_j = w$.

$\lambda_\ell(\lambda_t)$ onto T.

$\lambda_r(\lambda_t)$ onto T.

If T is empty then goto **termination** step.

Otherwise, repeat the **iteration** step.

3) Termination

The optimal path is $\mathbf{s}^* = s_1 s_2 \ldots s_L$ □

3.1.3 Computational Complexity

Let us examine the computational complexity of the alignment algorithm. The algorithm iterates for all $i = 1, \ldots, L-1$, $j = i+1, \ldots, L$ and $v = 2, \ldots, M-1$, $w = 2, \ldots, M-1$. The complexity of each **iteration** step depends on the type of the states v and w. Table 1 summarizes the computational complexity of each case of the **iteration** step of the alignment algorithm in Sec. 3.1.1. From this table, we can compute the total complexity of the alignment algorithm as follows

$$O(L^3M_1^2M) + O(L^2M_1M^2) + O(L^2M_2^2M) \tag{19}$$

Although the complexity in (19) is higher than $O(LM^2)$ of the Viterbi algorithm, it is still a polynomial in L and M, which is much more efficient than $O(LM^L)$ of the exhaustive search approach. The computational complexity of the alignment algorithm for general SCFGs in Chomsky normal form is $O(L^3M^3)$ [5, 23]. As we can see, the computational cost of both algorithms increases with $O(L^3M^3)$, in general. However, the csHMM usually requires less number of states for modeling sequences with certain correlations (e.g. palindrome language) than the SCFG in the *Chomsky normal form* (CNF) [14], hence it may have an computational advantage.

4 Computing the Probability of an Observed Sequence

Another important problem that arises in using HMMs for real-world applications is the following. Given an observation sequence $\mathbf{x} = x_1 \dots x_L$, how can we efficiently compute the probability $P(\mathbf{x}|\Theta)$ that this sequence was generated by the HMM with the set of parameters Θ? This is typically called the *scoring problem* for the following reason. Assume that we have K different models, each with different set of parameters $\Theta_k (k = 1, 2, \dots, K)$. Among these K HMMs, which one should we choose such that the probability of observing $\mathbf{x}$ is maximized? In order to choose the best model, we have to score each model based on the observation sequence $\mathbf{x}$, where the probability $P(\mathbf{x}|\Theta)$ is the natural choice for the score. Since $P(\mathbf{x}|\Theta)$ can be used for scoring different HMMs, the problem of computing this probability is called the scoring problem.

For regular HMMs, we can use the forward algorithm for solving this problem, whose complexity is the same as that of the Viterbi algorithm. However, due to the context-sensitive property of csHMMs, this algorithm cannot be directly used for scoring csHMMs. Even though the forward algorithm cannot be used for computing the probability $P(\mathbf{x}|\Theta)$ in context-sensitive HMMs, we can adopt a similar approach that was previously used in the optimal alignment algorithm. In Sec. 4.1, we propose a dynamic programming algorithm for scoring csHMM. In addition to this, we also propose the outside algorithm for csHMM in Sec. 4.2. This algorithm can be used together with the scoring algorithm for training context-sensitive HMMs, which will be elaborated in Sec. 5.

4.1 Scoring of csHMM

The csHMM scoring algorithm can be viewed as a variant of the alignment algorithm, where the *max* operators are replaced by *sums*. Conceptually, this algorithm is somewhat similar to the inside algorithm [23] that is used for scoring SCFGs. As in the alignment algorithm, we start from the inside of the observed symbol sequence and iteratively proceed to the outward direction. During this process, the pairwise-emission state P_n and the context-sensitive state C_n that interact with each other are considered at the same time.

In order to describe the algorithm, we use the same notations as in Sec. 3.1. In addition to this, we define the *inside variable* $\alpha(i, j, v, w)$ as the probability of all sub-paths $s_i \dots s_j$ with $s_i = v$ and $s_j = w$. It is assumed that all pairwise-emission states P_n inside the path are paired with the corresponding context-sensitive states C_n. Now, the scoring algorithm can be described as follows.

1) Initialization

For $i = 1, \ldots, L, v = 2, \ldots, M-1$.

$$\alpha(i, i, v, v) = \begin{cases} e(x_i|v) & v \in \mathcal{S} \\ 0 & \text{otherwise} \end{cases}$$

2) Iteration

For $i = 1, \ldots, L-1, j = i+1, \ldots, L$ and $v = 2, \ldots, M-1, w = 2, \ldots, M-1$.

(i) $v \in \mathcal{C}$ or $w \in \mathcal{P}$

$$\alpha(i, j, v, w) = 0$$

(ii) $v \in \mathcal{P}, w \in \mathcal{S}$

$$\alpha(i, j, v, w) = \sum_u \Big[\alpha(i, j-1, v, u) t(u, w) e(x_j|w)\Big]$$

(iii) $v \in \mathcal{S}, w \in \mathcal{C}$

$$\alpha(i, j, v, w) = \sum_u \Big[e(x_i|v) t(v, u) \alpha(i+1, j, u, w)\Big]$$

(iv) $v = P_n, w = C_m$ $(n \neq m), j < i+3$

$$\alpha(i, j, v, w) = 0$$

(v) $v = P_n, w = C_m$ $(n \neq m), j \geq i+3$

$$\alpha(i, j, v, w) = \sum_u \sum_{k=i+1}^{j-2} \alpha(i, k, v, \bar{v}) t(\bar{v}, u) \alpha(k+1, j, u, w)$$

(vi) $v = P_n, w = C_n, j = i+1$

$$\alpha(i, j, v, w) = e(x_i|v) t(v, w) e(x_j|w, x_i)$$

(vii) $v = P_n, w = C_n, j > i+1$

$$\begin{aligned} \alpha(i, j, v, w) &= \sum_u \sum_{k=i+1}^{j-2} \alpha(i, k, v, w) t(w, u) \alpha(k+1, j, u, w) \\ &\quad + \sum_{u_1} \sum_{u_2} \Big[e(x_i|v) t(v, u_1) \alpha(i+1, j-1, u_1, u_2) t(u_2, w) e(x_j|w, x_i)\Big] \end{aligned}$$

(viii) $v \in \mathcal{S}, w \in \mathcal{S}$

In this case, the variable $\alpha(i, j, v, w)$ can be updated using any of the update formulae in (ii) or (iii).

3) Termination

$$P(\mathbf{x}|\Theta) = \sum_v \sum_w t(1, v) \alpha(1, L, v, w) t(w, M) \qquad \square$$

At the end of the algorithm, we can obtain the probability $P(\mathbf{x}|\Theta)$ that the given csHMM will generate the observation sequence $\mathbf{x}$. The computational complexity of this algorithm is the same as the complexity of the alignment algorithm, which is shown in (19).

4.2 The Outside Algorithm

In a similar fashion, we can define the *outside variable* $\beta(i, j, v, w)$ to be the probability of all sub-paths $s_1 \dots s_i s_j \dots s_L$, where $s_i = v$ and $s_j = w$. In other words, $\beta(i, j, v, w)$ contains the probability of the entire sequence excluding $x_{i+1} \dots x_{j-1}$. This variable is needed for parameter re-estimation of csHMM, which will be elaborated in Sec. 5. As in Sec. 4.1, we assume that all pairwise-emission states P_n in $s_1 \dots s_i s_j \dots s_L$ are paired with the corresponding context-sensitive states C_n in a nested manner. Fig. 10 illustrates the state sequences that are considered in computing the variable $\beta(i, j, v, w)$, and the ones that are not taken into account.

In the outside algorithm, we start computing $\beta(i, j, v, w)$ from the outside of the sequence and proceed to the inward direction. As in the scoring algorithm, whenever there is an interaction between two symbols, the emission of these symbols are considered together. The inside variable $\alpha(i, j, v, w)$, which has been computed previously, is needed for computing the outside variable $\beta(i, j, v, w)$. Now, we can solve for $\beta(i, j, v, w)$ as follows.

1) Initialization

For $i = 1, \dots, L, v = 1, \dots, M$.

$$\beta(0, L+1, v, w) = \begin{cases} 1 & v = 1, w = M \\ 0 & \text{otherwise} \end{cases}$$

$$\beta(i, L+1, v, w) = \begin{cases} \sum_u t(1, u)\alpha(1, i, u, v) & w = M \\ 0 & \text{otherwise} \end{cases}$$

$$\beta(0, i, v, w) = \begin{cases} \sum_u \alpha(i, L, w, u)t(u, M) & v = 1 \\ 0 & \text{otherwise} \end{cases}$$

2) Iteration

For $i = 1, \dots, L-1, j = i+1, \dots, L$ and $v = 1, \dots, M, w = 1, \dots, M$.

(i) $v = 1$ or $w = M$

$$\beta(i, j, v, w) = 0$$

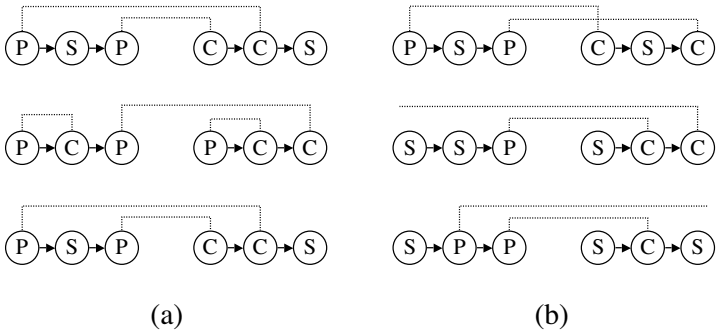

Figure 10: Examples of state sequences (a) that are considered in computing $\beta(i, j, v, w)$ and (b) those that are not considered.

(ii) $v \in \mathcal{P}$, $w \in \mathcal{P}$

$$\beta(i,j,v,w) \;=\; \sum_{u} \sum_{k=j+2}^{L+1} \beta(i,k,v,u)\alpha(j,k-1,w,\bar{w})t(\bar{w},u)$$

(iii) $v \in \mathcal{C}$, $w \in \mathcal{C}$

$$\beta(i,j,v,w) \;=\; \sum_{u} \sum_{k=0}^{i-2} \beta(k,j,u,w)\alpha(k+1,i,\bar{v},v)t(u,\bar{v})$$

(iv) $v \in \mathcal{C}$, $w \in \mathcal{P}$

$$\beta(i,j,v,w) \;=\; \sum_{u_1,u_2} \sum_{k_1=0}^{i-2} \sum_{k_2=j+2}^{L+1} \beta(k_1,k_2,u_1,u_2)\alpha(k_1+1,i,\bar{v},v) \times\alpha(j,k_2-1,w,\bar{w})t(u_1,\bar{v})t(\bar{w},u_2)$$

(v) $v \notin \mathcal{S}$, $w \in \mathcal{S}$

$$\beta(i,j,v,w) = \sum_{u} \beta(i,j+1,v,u)t(w,u)e(x_j|w)$$

(vi) $v \in \mathcal{S}$, $w \notin \mathcal{S}$

$$\beta(i,j,v,w) = \sum_{u} \beta(i-1,j,u,w)t(u,v)e(x_i|v)$$

(vii) $v = P_n, w = C_m (n \neq m)$

$$\beta(i,j,v,w) = 0$$

(viii) $v = P_n, w = C_n$

$$\beta(i,j,v,w) = \sum_{u_1,u_2} \beta(i-1,j+1,u_1,u_2)t(u_1,v)e(x_i|v)e(x_j|w,x_i)t(w,u_2)$$

(ix) $v \in \mathcal{S}, w \in \mathcal{S}$

In this case, the variable $\alpha(i,j,v,w)$ can be updated using either (v) or (vi).

3) Termination

$$P(\mathbf{x}|\Theta) \;=\; \sum_{v,w} \beta(i,i+1,v,w)t(v,w) \quad \text{for any } i \qquad \square$$

Let us first look at the **initialization** step. For the case of an empty string, i.e. when $i = 0$ and $j = L+1$, we set $\beta(0, L+1, 1, M)$ to unity. When $i \geq 1$ and $j = L+1$, all the pairwise interactions have to occur inside the sub-path $s_1 \ldots s_i$. Since $\alpha(1,i,u,v)$ is the probability of all sub-paths for $x_1 x_2 \ldots x_i$ with $s_1 = u$ and $s_i = v$, we can compute $\beta(i, L+1, v, M)$ by taking the product of the transition probability from state 1 to state u and the inside variable $\alpha(1,i,u,v)$, and then adding this product over all u. The case when $i = 0$ and $j \leq L$ can be treated similarly. These are shown in the **initialization** step.

After the initialization of the outside variable $\beta(i,j,v,w)$, we proceed into the **iteration** step. Firstly, consider the case when $v \in \mathcal{P}$ and $w \in \mathcal{P}$. Since all pairwise-emission states have to be paired with the corresponding context-sensitive states in a nested manner, $s_j = w$ has to pair with $\bar{w}$ between $j+1$ and

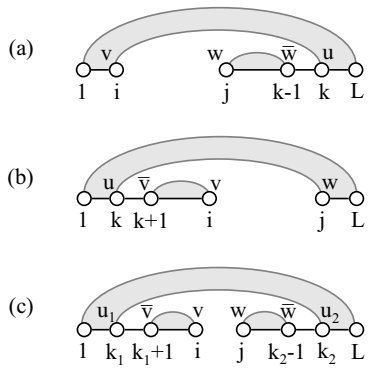

Figure 11: Illustration of the iteration step of the outside algorithm. (a) Case (ii). (b) Case (iii). (c) Case (iv).

$k-1$ as shown in Fig. 11 (a). As in Fig. 8 and Fig. 9, the shaded regions indicate that all P_n and C_n states are paired inside each region. Similarly, $s_i = v$ has to form a pair with $\bar{v}$ between k and L, and all the interactions in the sub-path $x_1 \dots x_i x_k \dots x_L$ should be paired in a nested manner. Since the probability of each sub-path $s_j \dots s_{k-1}$ and $s_1 \dots s_i s_k \dots s_L$ is contained in $\alpha(j, k-1, w, \bar{w})$ and $\beta(i, k, v, u)$ respectively, we can compute $\beta(i, j, v, w)$ as described in (ii) of the **iteration** step. Fig. 11 (b) illustrates the case when $v \in \mathcal{C}$ and $w \in \mathcal{C}$. In this case, $\beta(i, j, v, w)$ can be updated in a similar manner as shown in (iii). Fig. 11 (c) shows the case when $v \in \mathcal{C}$ and $w \in \mathcal{P}$. As shown in the figure, $s_i = v$ has to pair with $\bar{v}$ between $k_1 + 1$ and $i - 1$ and $s_j = w$ also has to pair with $\bar{w}$ between $j + 1$ and $k_2 - 1$. All the other interactions have to be confined within the state sequence $s_1 \dots s_{k_1} s_{k_2} s_L$. Therefore, $\beta(i, j, v, w)$ can be computed as in (iv) of the **iteration** step.

When w is a single-emission state, $\beta(i, j, v, w)$ can be obtained simply by extending $\beta(i, j+1, v, u)$ by one sample, as depicted in Fig. 12 (a). As shown in (v) of the **iteration** step, we first compute the product of $\beta(i, j+1, v, u)$ and the transition probability $t(w, u)$ and the emission probability of the symbol x_j at the state $s_j = w$, and add the product over u. $\beta(i, j, v, w)$ can be computed likewise when $v \in \mathcal{S}$, as described in (vi). If both v and w are single-emission states, we may use either (v) or (vi) for updating the outside variable $\beta(i, j, v, w)$. Finally, let us consider the case when $v = P_n$ and $w = C_m$. Since there

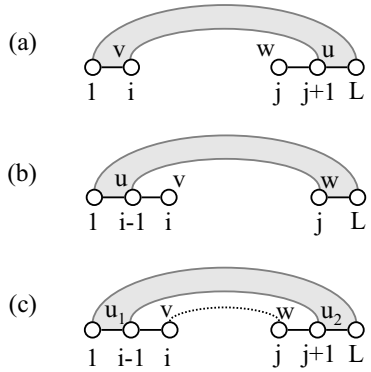

Figure 12: Illustration of the iteration step of the outside algorithm. (a) Case (v). (b) Case (vi). (c) Case (viii).

can be no crossing interactions, $s_i = P_n$ and $s_j = C_m$ have to interact with each other, as illustrated in Fig. 12 (c). The dotted line indicates the pairwise interaction between x_i and x_j. For this reason, n has to be the same as m, and $\beta(i, j, v, w)$ is set to zero if $n \neq m$. For $n = m$, we can compute $\beta(i, j, v, w)$ by extending $\beta(i-1, j+1, u_1, u_2)$ as shown in (viii) of the **iteration** step.

Once the **iteration** step is complete, the **termination** step of the outside algorithm also yields the probability $P(\mathbf{x}|\Theta)$ like the scoring algorithm in Sec. 4.1. The computational complexity of the outside algorithm is usually not an issue, since it is mainly used for training the model offline.

5 Re-estimation of Model Parameters

In order to apply context-sensitive HMMs to real-world problems, it is crucial to adjust the model parameters in an optimal way. Therefore, it is important to find a method for optimizing the set of model parameters Θ, such that the probability $P(\mathbf{x}|\Theta)$ of the given observation sequence $\mathbf{x}$ is maximized. The process of finding these optimal parameters is typically called "training". Although it is infeasible to find an analytical solution for the optimal parameters, we can use the EM (expectation-maximization) approach for finding parameters that achieve a local maximum of $P(\mathbf{x}|\Theta)$. In traditional HMMs, Baum-Welch algorithm [16] has been widely used for iterative update of the parameters. Similarly, there exists an EM algorithm, called the inside-outside algorithm [23], which can be used for optimizing the model parameters of a SCFG. Both algorithms compute an estimate $\widehat{\Theta}$ of the model parameters based on the given observation sequence and the current set of parameters Θ. The current set of model parameters Θ is then updated by this estimate $\widehat{\Theta}$, and this re-estimation procedure is repeated until a certain stopping criterion is satisfied.

A similar approach can also be used for iterative re-estimation of the model parameters in a context-sensitive HMM. In order to describe the re-estimation algorithm, let us first define the following variables.

$$
\begin{aligned}
\tau_i(v, w) &= \text{The probability that } s_i = v \text{ and } s_{i+1} = w \text{ given the model } \Theta \text{ and the} \\
&\quad \text{observed symbol string } \mathbf{x} \\
\sigma_i(v) &= \text{The probability that } s_i = v \text{ given } \Theta \text{ and } \mathbf{x} \\
\delta_v(i, j) &= \text{The probability that } s_i = v \text{ and } s_j = \bar{v} \text{ have an interaction with each other}
\end{aligned}
$$

Firstly, $\tau_i(v, w)$ can be computed as follows

$$\tau_i(v, w) = \frac{\beta(i, i+1, v, w) t(v, w)}{P(\mathbf{x}|\Theta)}. \tag{20}$$

The probability $\sigma_i(v)$ can be obtained simply by adding $\tau_i(v, w)$ over all w

$$\sigma_i(v) = \sum_w \tau_i(v, w). \tag{21}$$

Finally, the probability $\delta_v(i, j)$ can be written as

$$\delta_v(i, j) = \frac{\sum_{u_1, u_2} \alpha(i, j, v, \bar{v}) \beta(i-1, j+1, u_1, u_2) t(u_1, v) t(\bar{v}, u_2)}{P(\mathbf{x}|\Theta)}. \tag{22}$$

Based on these probabilities, we can compute the expected number of occurrences of a state v in the path as well as the number of transitions from a state v to another state w. For example, if we add $\tau_i(v, w)$ over all locations i, we get

$$\sum_{i=0}^{L} \tau_i(v, w) = \text{Expected number of transitions from } v \text{ to } w. \tag{23}$$

Similarly, if we add $\sigma_i(v)$ over all i, we obtain the following

$$\sum_{i=0}^{L} \sigma_i(v) = \text{Expected number of transitions from } v. \tag{24}$$

Now, we can re-estimate the model parameters of the csHMM using the following method. To begin with, let us first compute an estimate of the transition probability from v to w, where $v \in \mathcal{P}$ or $v \in \mathcal{S}$. In this case, the estimate is given by

$$\begin{aligned} \hat{t}(v, w) &= \frac{\text{Expected number of transitions from } v \text{ to } w}{\text{Expected number of transitions from } v} \\ &= \frac{\sum_{i=0}^{L} \tau_i(v, w)}{\sum_{i=0}^{L} \sigma_i(v)}. \end{aligned} \tag{25}$$

For $v = C_n$, the set of states to which v can make a transition differs depending on whether the corresponding stack is empty or not. If $w \in \mathcal{E}_n$, i.e. if w is a state to which $v = C_n$ can make a transition when the stack is empty,

$$\begin{aligned} \hat{t}(v, w) &= \frac{\text{Expected number of transitions from } v = C_n \text{ to } w \in \mathcal{E}_n}{\text{Expected number of transitions from } v = C_n \text{ to any state in } \mathcal{E}_n} \\ &= \frac{\sum_{i=0}^{L} \tau_i(v, w)}{\sum_{i=0}^{L} \sum_{u \in \mathcal{E}_n} \tau_i(v, u)}. \end{aligned} \tag{26}$$

If $w \in \mathcal{F}_n$, then we can obtain the estimate by

$$\begin{aligned} \hat{t}(v, w) &= \frac{\text{Expected number of transitions from } v = C_n \text{ to } w \in \mathcal{F}_n}{\text{Expected number of transitions from } v = C_n \text{ to any state in } \mathcal{F}_n} \\ &= \frac{\sum_{i=0}^{L} \tau_i(v, w)}{\sum_{i=0}^{L} \sum_{u \in \mathcal{F}_n} \tau_i(v, u)}. \end{aligned} \tag{27}$$

Now, let us estimate the emission probability $e(x|v)$ and $e(x|v, x_p)$. For $v \in \mathcal{P}$ or $v \in \mathcal{S}$, the emission probability does not depend on the context. Therefore, we can compute the estimate $\hat{e}(x|v)$ of the emission probability as follows

$$\begin{aligned} \hat{e}(x|v) &= \frac{\text{Expected number of times that the symbol } x \text{ was emitted at state } v}{\text{Expected number of occurrences of state } v} \\ &= \frac{\sum_{i=1|x_i=x}^{L} \sigma_i(v)}{\sum_{i=1}^{L} \sigma_i(v)} \end{aligned} \tag{28}$$

In contrast, if v is a context-sensitive state, the emission probability is dependent on the symbol x_p that was emitted at the corresponding pairwise-emission state $\bar{v}$. Bearing this in mind, we can estimate $\hat{e}(x|v, x_p)$

as follows.

$$\begin{aligned}\hat{e}(x|v,x_p) &= \frac{\text{Expected number of emissions of } x \text{ at state } v\in\mathcal{C} \text{ given the context } x_p}{\text{Expected number of emissions at state } v\in\mathcal{C} \text{ given the context } x_p} \\ &= \frac{\sum_{j=2|x_j=x}^{L}\sum_{i=1|x_i=x_p}^{j-1}\delta_v(i,j)}{\sum_{j=2}^{L}\sum_{i=1|x_i=x_p}^{j-1}\delta_v(i,j)} \end{aligned} \tag{29}$$

Although we derived these update formulae based on a single observation sequence $\mathbf{x}$, they can be easily extended for multiple training sequences. When we have more than one observation sequence for training, we simply add all the expected counts over all sequences, and use these numbers for estimating the model parameters.

Now that we have the estimates $\hat{t}(v,w)$, $\hat{e}(x|v)$ and $\hat{e}(x|v,x_c)$, we can update the model parameters by these estimates

$$\begin{aligned} t(v,w) &\longleftarrow \hat{t}(v,w) \\ e(x|v) &\longleftarrow \hat{e}(x|v) \\ e(x|v,x_p) &\longleftarrow \hat{e}(x|v,x_p). \end{aligned}$$

We repeat this re-estimation procedure until a certain stopping criterion is satisfied. As mentioned earlier, the training of the model is performed offline, and therefore the computational cost of the re-estimation algorithm is usually not a critical issue.

6 Simulation Results

In order to test the proposed algorithms, let us consider the example in Fig. 13. This csHMM generates sequences with long-range correlations between distant symbols. Such pairwise dependencies are commonly found in the so-called "iron response elements" in RNA sequences [24]. The model in Fig. 13 has three single-emission states S_1, S_2 and S_3, and two pairs of pairwise-emission states and context-sensitive states. Each pair (P_1, C_2) and (P_2, C_2) is associated with a separate stack. The transition probabilities are shown in Fig. 13 along the edges. Each state emits one of the four symbols $\mathcal{A} = \{A, C, G, U\}$, where the emission probabilities are as shown in Table 2. Every row in Table 2 contains the emission probabilities that each output symbol will be emitted at the given state. For example, the first row in Table 2 shows the probabilities that the symbols A, C, G, and U will be emitted at P_1. Therefore, each row adds up to unity. The emission probabilities at C_n are dependent on the symbol x that was emitted at the corresponding

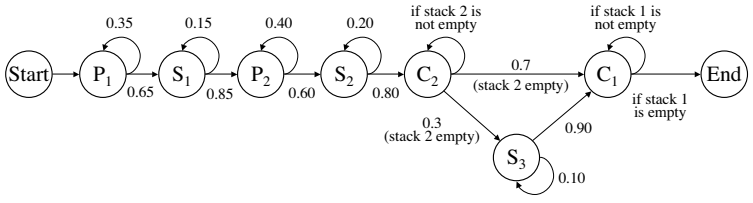

Figure 13: An example of a context-sensitive HMM.

	A	C	G	U
P_1	0.55	0.05	0.05	0.35
S_1	0.15	0.35	0.35	0.15
P_2	0.40	0.05	0.05	0.50
S_2	0.05	0.60	0.30	0.05
S_3	0.05	0.10	0.75	0.10

Table 2: Emission probabilities $e(x|v)$.

state P_n. In this example, we set the emission probabilities of C_1 and C_2 such that they always emit the "complementary" symbol of x ($A \leftrightarrow U$ and $C \leftrightarrow G$ are complementary to each other).

Now, let us assume that the observed symbol string is $\mathbf{x} = AUCUACUAAU$. What is the optimal state sequence $\mathbf{s}^* = s_1 s_2 \dots s_{10}$ that maximizes the probability of observing $\mathbf{x}$ based on the specified model? Using the alignment algorithm elaborated in Sec. 3, we obtained

$$\mathbf{s}^* = P_1 P_1 S_1 P_2 P_2 S_2 C_2 C_2 C_1 C_1, \tag{30}$$

where the log-probability of $\mathbf{s}^*$ was $\log_2 P(\mathbf{x}, \mathbf{s}^*|\Theta) = -12.2165$. In order to check the validity of this result, we performed an exhaustive search over all possible paths. Since the length of the sequence is $L = 10$, and as there are $M - 2 = 7$ emitting states, we have $(M-2)^L = 7^{10} = 282,475,249$ possibilities. By comparing the log-probabilities of all paths, we obtained the same optimal path as (30) with the same log-probability, which shows that the optimal alignment algorithm works as expected. Similarly, we computed the probability of the sequence $\mathbf{x}$, given the model in Fig. 13. Using the scoring algorithm in Sec. 4, we obtained

$$P(\mathbf{x}|\Theta) = 2.1146 \times 10^{-4}. \tag{31}$$

Again, we computed the probability using the brute-force approach by considering all possible paths and adding the probability of each path. As a result, we obtained

$$P(\mathbf{x}|\Theta) = \sum_{\mathbf{s}} P(\mathbf{x}, \mathbf{s}|\Theta) = 2.1146 \times 10^{-4}, \tag{32}$$

which is the same as (31). As we can see from these results, the proposed scoring and alignment algorithms are capable of finding the same solutions as the brute-force methods in a much more efficient manner.

Now, let us consider the training of the csHMM. In order to test the parameter re-estimation algorithm, we first generated 200 symbol sequences based on the model in Fig. 13. Then, we randomly initialized the transition and emission probabilities of the model, and ran the algorithm in Sec. 5 to optimize the model parameters. Fig. 14 shows the arithmetic mean and the geometric mean of the sequence probabilities after each iteration. As we can see, the mean values are nearly zero in the beginning, since the parameters have been randomly initialized. The model parameters quickly converged to the final values after only a few iterations, and the converged values were very close to the original values. Table 3 shows the estimated emission probabilities after 10 iterations. By comparing it with Table 2, we can see that the estimated values are close to the original ones.

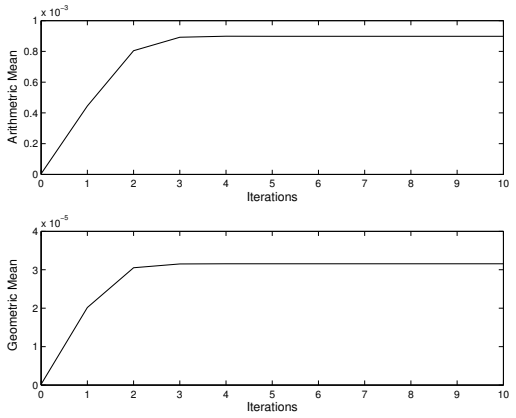

Figure 14: The arithmetic mean (top) and the geometric mean (bottom) after each iteration.

	A	C	G	U
P_1	0.5503	0.0617	0.0348	0.3533
S_1	0.1258	0.4094	0.3481	0.1167
P_2	0.4067	0.0628	0.0400	0.4905
S_2	0.0477	0.5543	0.3528	0.0452
S_3	0.0870	0.1364	0.7073	0.0693

Table 3: Estimated emission probabilities $e(x|v)$ after 10 iterations.

7 Discussion

As we have seen, context-sensitive HMMs can be effectively used for modeling pairwise interactions between distant symbols in a symbol string. In this section, we consider possible extensions of the basic model and discuss several interesting issues regarding the csHMM.

7.1 Emission of multiple symbols

In this paper, we assumed that every state in the csHMM emits only one symbol at a time. Based on this assumption, we considered only sequences with *pairwise* dependencies between distant symbols that are arranged in a nested manner. However, we can easily extend the basic model such that it can also describe *non-pairwise* dependencies, by allowing the states to emit two or more symbols at a time. For example, we may modify the model in Fig. 4 such that the context-sensitive state C_1 emits two symbols at a time. When we enter C_1, the symbol x that is on the top of the stack is popped out, and the emission probabilities of C_1 are adjusted so that it emits xx. In this way, the modified model will generate sequences of the form $x_1x_2 \ldots x_Nx_Nx_N \ldots x_2x_2x_1x_1$. An example of such a symbol sequence is shown in Fig. 15. As shown in this figure, the correlations still occur in a nested manner, but they are not limited to pairwise

Figure 15: An example sequence that can be generated by the modified model of Fig. 4.

correlations any more. Such modifications can be easily incorporated into the algorithms described in the previous sections. For example, we may change the second term in the update formula (vii) in Sec. 4.1 to

$$\sum_{u_1}\sum_{u_2}\Big[e(x_i\dots x_{i+\delta_n^p-1}|v)t(v,u_1)\alpha(i+\delta_n^p,j-\delta_n^c,u_1,u_2) \\ \times t(u_2,w)e(x_{j-\delta_n^c+1}\dots x_j|w,x_i\dots x_{i+\delta_n^p-1})\Big], \tag{33}$$

when the csHMM is modified such that the pairwise-emission state P_n emits δ_n^p symbols at a time and the corresponding context-sensitive state C_n emits δ_n^c symbols at a time.

7.2 Modeling crossing correlations

Although we have mainly focused on context-sensitive HMMs that generate sequences with nested correlations, the descriptive power of the proposed model is not restricted to such a correlation structure. In fact, csHMM can be used to represent sequences with various correlations between symbols, including crossing dependencies. Fig. 16 shows an example of such a csHMM. Note that the csHMM in Fig. 16 still uses stacks, but the P_n and C_n states are arranged such that the model gives rise to crossing interactions between symbols. Furthermore, we may also replace the *stack* by a *queue* to represent other types of interactions. For example, we can describe the *copy language* by using a csHMM with a queue. The copy language includes all sequences that consist of the concatenation of two identical sequences. The model illustrated in Fig. 17 can effectively represent such a language. When the given csHMM generates sequences with crossing interactions, the algorithms in Sec. 3, Sec. 4, and Sec. 5 cannot be directly used. However, it is possible to extend the proposed algorithms such that they can be used for csHMMs with crossing interactions as those shown in Fig. 16 and Fig. 17. For example, for scoring such csHMMs, we may define the variable $\alpha(i,j,k,\ell,u,v,w,x)$ as the probability of the sub-sequence $x_i\dots x_j x_k\dots x_\ell$ $(i\le j<k\le\ell)$, where $s_i=u, s_j=v, s_k=w, s_\ell=x$ and all P_n states are paired with the corresponding C_n states inside the sub-path $s_i\dots s_j s_k\dots s_\ell$. We can compute $\alpha(\dots)$ in a recursive manner by considering crossing correlations between s_i and s_k, s_j and s_ℓ, and so forth. This is illustrated in Fig. 18. In this case, the computational complexity of the algorithm will be considerably higher than $O(L^3M^3)$.

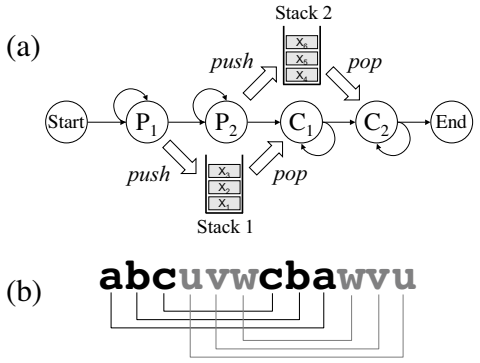

Figure 16: (a) A csHMM that results in crossing interactions. (b) An example of a generated sequence. The lines indicate the correlations between symbols.

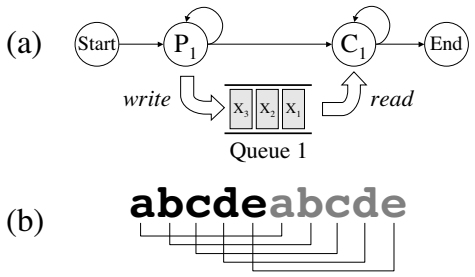

Figure 17: (a) A csHMM that represents a copy language. (b) An example of a generated sequence.

7.3 Comparison with other variants of HMM

As mentioned earlier, there exist many interesting variants of the traditional HMM, which extend the basic model in various ways [7]-[13]. For example, the hidden semi-Markov model (HSMM) allows us to associate an explicit state occupancy distribution with each state [7]-[11], instead of using the implicit geometric state occupancy distribution in the basic HMM. However, the hidden states in the HSMM are not context-sensitive, and the emission and transition probabilities of the future states do not explicitly depend on the symbols that have been emitted previously. Therefore, these models cannot explicitly model pairwise correlations between distant symbols as the csHMM does.

There exists another interesting generalization of the HMM called the pairwise Markov chain (PMC) [12]. The PMC assumes that the pair of the random variables (x_i, s_i) is a Markov chain. This model is mathematically more general than the HMM, which is a special case of the PMC, where the hidden state s_i satisfies the Markov property. Since the pair (x_i, s_i) is a Markov chain, the probabilities associated with x_i, s_i, and (x_i, s_i) do not depend on the previous emissions, and the PMC cannot be used for describing complex correlations such as the ones observed in palindromes. This is also the case with the triplet Markov chain (TMC) [13], which is a further generalization of the PMC, and there exists a fundamental difference between the csHMM and the PMC/TMC.

7.4 Comparison with other stochastic grammars

As HMMs are equivalent to stochastic regular grammars (SRG), the csHMM can be viewed as an extension of the SRG with specific context-sensitive production rules. Therefore, the SRG is a proper subset of the proposed csHMM. The context-sensitive property of the csHMM enables the model to describe explicit dependencies between distant symbols, which are beyond the descriptive power of SRGs. As a result, the csHMM is capable of modeling sequences with nested correlations, which are characteristic of languages that are described by SCFGs. This implies that the csHMM can be used as a good alternative to SCFGs,

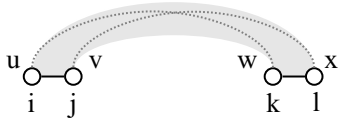

Figure 18: An illustration of the basic concept of the algorithm that can be used when there exist crossing interactions. The dotted lines show examples of correlations that can be taken into consideration based on this setting.

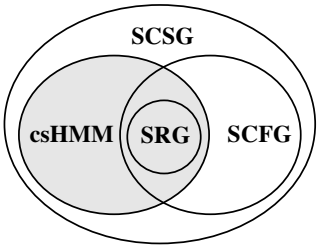

Figure 19: The csHMM in the Chomsky hierarchy.

in many practical situations. Moreover, the csHMM is also capable of modeling crossing correlations as illustrated in the examples shown in Fig. 16 and Fig. 17. This cannot be done using a SCFG, and we have to resort to higher order grammars such as the stochastic context-sensitive grammars (SCSG). However, there exist also languages that can be described by a context-free grammar but not by a csHMM. One such example can be found in the appendix. This shows that even though there is a considerable overlap between csHMMs and SCFGs, neither of them fully includes the other. Finally, the csHMM can be viewed as a stochastic formal grammar that uses only non-contracting production rules.[4] It is known that for any non-contracting grammar there exists an equivalent context-sensitive grammar [22]. This implies that the csHMM is a subset of the stochastic context-sensitive grammars (SCSG). The full relationship between the csHMM and other stochastic grammars is illustrated in the Venn diagram shown in Fig. 19.

The capability of modeling various correlations (including nested and/or crossing interactions) based on a single framework is a significant advantage of csHMMs over SRGs and SCFGs. Another advantage of the proposed model is that it can *explicitly* describe the dependencies between distant symbols. This allows us to model the symbol sequences of our interest in a simple and a direct way, which can be an advantage (although arguable) compared to the SCFGs, unless a tree-structured design is preferred for some reason. When modeling sequences with crossing interactions, this capability stands out more prominently. Although the SCSGs can represent sequences with crossing interactions, they cannot directly generate the crossing interactions in the symbol sequence. For example, when modeling the *copy language*, the crossing dependencies between symbol pairs cannot be directly generated [5]. Instead, the grammar generates the two related non-terminals in a non-crossing manner, and applies the context-sensitive re-ordering rules later on, in order to obtain the final sequence that has crossing correlations. For this reason, context-sensitive grammars can be quite complex even for simple languages.

8 Conclusion

In this paper, we have introduced the idea of context-sensitive HMMs. They can be viewed as an extension of the traditional HMM, where some of the states are equipped with auxiliary memory. Symbols that are emitted at certain states are stored in this memory, and the stored data serves as the context of the system, which affects the emission probabilities and the transition probabilities of the model. In this way, we can represent long-range interactions between distant symbols, which cannot be done using traditional HMMs. The csHMM is a very efficient tool for modeling sequences with complex dependencies, and it

[4]This means that none of the production rules decrease the length of the symbol string [22].

can be used as a good alternative to stochastic grammars such as the SCFG and the SCSG. We also proposed efficient polynomial-time algorithms for finding the optimal state sequence and for computing the probability of an observed symbol string. These algorithms can be used for solving the alignment problem and the scoring problem of context-sensitive HMMs with nested interactions. Furthermore, a parameter re-estimation algorithm has been introduced, which can be used for training a csHMM based on a number of training sequences. The proposed model has an interesting application in Bioinformatics, especially in RNA sequence analysis [21].

9 Appendix

In the following, we give an example of a language that can be described by a context-free grammar but not by a csHMM. Let us consider a context-free grammar that has two non-terminal symbols S, T and three terminal symbols a, b, c. We begin with the start non-terminal S and apply the the following production rules

$$\begin{aligned} S &\longrightarrow aTSa \mid TaSa \mid TSaa \mid aTa \mid Taa \\ T &\longrightarrow bT \mid bc \end{aligned}$$

The grammar shown above can generate any sequence $\mathbf{x} = x_1 \dots x_{L-N} x_{L-N+1} \dots x_L$ for any given positive number N, where $x_{L-N+1} \dots x_L = a \dots a$ and $x_1 \dots x_{L-N}$ contains N number of 'a's and the same number of subsequences in the form of '$b \dots bc$'. For example, we can generate the following sequences using this grammar

$$(N=3) \qquad \underbrace{a\ bbbc\ a\ bc\ a\ bbc}_{x_1 \dots x_{L-N}} \ \underbrace{aaa}_{x_{L-N+1} \dots x_L} \tag{34}$$

$$\underbrace{bbc\ bc\ a\ a\ a\ bbbbc}_{x_1 \dots x_{L-N}} \ \underbrace{aaa}_{x_{L-N+1} \dots x_L} \tag{35}$$

$$(N=4) \quad \underbrace{bc\ a\ bbbbc\ a\ bbc\ a\ bc\ a}_{x_1 \dots x_{L-N}} \ \underbrace{aaaa}_{x_{L-N+1} \dots x_L} \tag{36}$$

$$\underbrace{bbc\ bc\ bbbc\ bbc\ a\ a\ a\ a}_{x_1 \dots x_{L-N}} \ \underbrace{aaaa}_{x_{L-N+1} \dots x_L} \tag{37}$$

It is *not* possible to construct a csHMM that generates *only* sequences in the above form. This can be seen from the following. As shown in the above examples, the number of 'a's in the tail $x_{L-N+1} \dots x_L$ is always identical to the number of 'a's and the number of subsequences '$b \dots bc$' in the head part $x_1 \dots x_{L-N}$. As the transition probabilities at single-emission states S_n and pairwise-emission states P_n do not depend on past emissions, the only way to ensure the generation of specific number of 'a's in the tail is to use context-sensitive states C_n, which have variable transition probabilities that depend on the context. As the last N symbols are emitted at context-sensitive states, identical number of symbols in $x_1 \dots x_{L-N}$ have to be emitted at the corresponding pairwise-emission states. Since the number of 'a's and the number of '$b \dots bc$' in the head part are both N, we may consider the following two cases. Firstly, we may consider using the corresponding pairwise-emission states to generate the 'a's in the head part. As the emitted symbols at these states are used as the context for generating identical number symbols in the tail, the

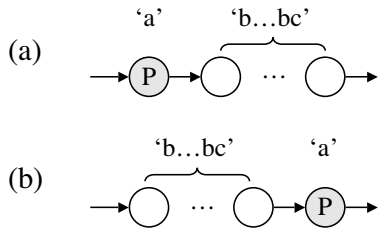

Figure 20: Constructions which guarantee that the number of 'a's and the number of '$b \ldots bc$' in $x_1 \ldots x_{L-N}$ are identical.

subsequences '$b \ldots bc$' cannot make use of this context. Therefore, the only way to guarantee that the number of '$b \ldots bc$' are also N is to construct the csHMM such that the states that generate '$b \ldots bc$' always follow (or precede) the pairwise-emission states that generate 'a's. This is illustrated in Fig. 20. Although this construction guarantees that the number of 'a's and the number of '$b \ldots bc$' in $x_1 \ldots x_{L-N}$ are both N, it cannot give rise to all possible orders of 'a's and '$b \ldots bc$'s. For example, such a csHMM cannot generate sequences in (35) and (37). Similar reasoning also holds when the pairwise-emission states, which correspond to the context-sensitive states used for generating the tail part, are used to generate (part of) the subsequence '$b \ldots bc$'. This leads to the conclusion that a construction which guarantees the emission of N 'a's and '$b \ldots bc$' cannot generate sequences such as (35) and (37). Therefore, the given context-free language cannot be represented by a csHMM.

10 Acknowledgment

We would like to thank the anonymous reviewers for their insightful remarks and valuable suggestions, which have been very helpful in improving the paper.

References

[1] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition", *Proceedings of the IEEE*, vol. 77, pp. 257-286, 1989.

[2] F. Jelinek, *Statistical methods for speech recognition*, The MIT Press, Cambridge, MA, 2001.

[3] A. Krogh, I. Saira Mian, D. Haussler, "A hidden Markov model that finds genes in E. coli DNA", *Nucleic Acids Res.*, vol. 22, pp. 4768-4778, 1994.

[4] S. L. Salzberg, A. L. Delcher, S. Kasif, O. White, "Microbial gene identification using interpolated Markov models", *Nucleic Acids Res.*, vol. 26, pp. 544-548, 1998.

[5] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, *Biological sequence analysis*, Cambridge Univ. Press, Cambridge, UK, 1998.

[6] S. R. Eddy, "Multiple alignment using hidden Markov models", *Proc. of the Third International Converence on Intelligent Systems for Molecular Biology*, pp. 112-120, 1995.

[7] Y. Guédon, "Estimating hidden semi-Markov chains from discrete sequences", *J. Comp. Graphical Stat.*, vol. 12, no. 3, pp. 604-639, 2003.

[8] Y. Guédon, "Hidden hybrid Markov/semi-Markov chains", *Computational Statistics & Data Analysis*, vol. 49, no. 3, pp. 663-688, 2005.

[9] S. Faisan, L. Thoraval, J.-P. Armspach, M.-N. Metz-Lutz, and F. Heitz, "Unsupervised learning and mapping of active brain functional MRI signals based on hidden semi-Markov event sequence models", *IEEE Transactions on Medical Imaging*, vol. 24, no. 2, pp. 263-276., 2005.

[10] S.-Z. Yu and H. Kobayashi, "An efficient forward-backward algorithm for an explicit-duration hidden Markov model", *IEEE Signal Processing Letters*, vol. 10, no. 1, pp. 11-14, 2003.

[11] M. D. Moore and M. I. Savic, "Speech reconstruction using a generalized HSMM (GHSMM)", *Digital Signal Processing*, vol. 14, no. 1, pp. 37-53, 2004.

[12] W. Pieczynski, "Pairwise Markov chains", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, 2003.

[13] W. Pieczynski, C. Hulard, and T. Veit, "Triplet Markov chains in hidden signal restoration", *Proc. SPIE's International Symposium on Remote Sensing*, Crete, Greece, Sep. 2002.

[14] N. Chomsky, "On certain formal properties of grammars", *Information and Control*, vol. 2, pp. 137-167, 1959.

[15] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm", *IEEE Transactions on Information Theory*, IT-13, pp. 260-267, 1967.

[16] L. E. Baum, "An equality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes", *Inequalities*, vol. 3, pp. 1-8, 1972.

[17] R. Schwartz, J. Klovstadt, L. Makhoul, and J. Sorensen, "Improved hidden Markov modeling of phonemes for continuous speech recognition", *Proc. International Conference on Acoustics, Speech and Signal Processing*, pp. 872-875, Denver, 1980.

[18] K. Lee, "Context-dependent phonetic hidden Markov models for speaker-independent continuous speech recognition", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38 (4), pp. 599-609, April 1990.

[19] A. Ljolje, "High accuracy phone recognition using context clustering and quasi-triphonic models", *Computer Speech & Language*, vol. 8, pp. 129-151, April 1994.

[20] Byung-Jun Yoon and P. P. Vaidyanathan, "HMM with auxiliary memory: A new tool for modeling RNA secondary structures", *Proc. 38th Asilomar Conference on Signals, Systems, and Computers*, Monterey, CA, Nov. 2004.

[21] Byung-Jun Yoon and P. P. Vaidyanthan, "An overview of the role of context-sensitive HMMs in the prediction of ncRNA genes", *Proc. IEEE Workshop on Statistical Signal Processing*, Bordeaux, France, July 2005.

[22] M. A. Harrison, *Introduction to formal language theory*, Addison-Wesley, 1978.

[23] K. Lari and S. J. Young, "The estimation of stochastic context-free grammars using the inside-outside algorithm", *Computer Speech and Language*, vol. 4, pp. 35-56, 1990.

[24] M. W. Hentze and L. C. Kuhn, "Molecular control of vertebrate iron metabolism: mRNA-basedregulatory circuits operated by iron, nitric oxide, and oxidativestress", *Proc. Natl. Acad. Sci., vol.93*, pp. 8175-8182, 1996.

Byung-Jun Yoon (S'02) was born in Seoul, Korea in 1975. He received the B.S.E. (summa cum laude) degree from Seoul National University (SNU), Seoul, Korea in 1998 and the M.S. degree from California Institute of Technology (Caltech), Pasadena, CA in 2002, both in electrical engineering. He is currently a Ph.D. candidate in electrical engineering at Caltech. His research interests include multirate signal processing and applications, bioinformatics and genomic signal processing. He received the Killgore Fellowship in 2001 from Caltech, and he was selected as a recipient of the Microsoft Research Graduate Fellowship for the year of 2004-2005. In 2003, he was awarded a prize in the student paper contest in the 37th Asilomar conference on signals, systems, and computers.

P. P. Vaidyanathan (S'80–M'83–SM'88–F'91) was born in Calcutta, India on Oct. 16, 1954. He received the B.Sc. (Hons.) degree in physics and the B.Tech. and M.Tech. degrees in radiophysics and electronics, all from the University of Calcutta, India, in 1974, 1977 and 1979, respectively, and the Ph.D degree in electrical and computer engineering from the University of California at Santa Barbara in 1982. He was a post doctoral fellow at the University of California, Santa Barbara from Sept. 1982 to March 1983. In March 1983 he joined the electrical engineering department of the Calfornia Institute of Technology as an Assistant Professor, and since 1993 has been Professor of electrical engineering there. His main research interests are in digital signal processing, multirate systems, wavelet transforms and signal processing for digital communications.

Dr. Vaidyanathan served as Vice-Chairman of the Technical Program committee for the 1983 IEEE International symposium on Circuits and Systems, and as the Technical Program Chairman for the 1992 IEEE International symposium on Circuits and Systems. He was an Associate editor for the IEEE Transactions on Circuits and Systems for the period 1985-1987, and is currently an associate editor for the journal IEEE Signal Processing letters, and a consulting editor for the journal Applied and computational harmonic analysis. He has been a guest editor in 1998 for special issues of the IEEE Trans. on Signal Processing and the IEEE Trans. on Circuits and Systems II, on the topics of filter banks, wavelets and

subband coders. Dr. Vaidyanathan has authored a number of papers in IEEE journals, and is the author of the book Multirate systems and filter banks. He has written several chapters for various signal processing handbooks. He was a recepient of the Award for excellence in teaching at the California Institute of Technology for the years 1983-1984, 1992-93 and 1993-94. He also received the NSF's Presidential Young Investigator award in 1986. In 1989 he received the IEEE ASSP Senior Award for his paper on multirate perfect-reconstruction filter banks. In 1990 he was recepient of the S. K. Mitra Memorial Award from the Institute of Electronics and Telecommuncations Engineers, India, for his joint paper in the IETE journal. He was also the coauthor of a paper on linear-phase perfect reconstruction filter banks in the IEEE SP Transactions, for which the first author (Truong Nguyen) received the Young outstanding author award in 1993. Dr. Vaidyanathan was elected Fellow of the IEEE in 1991. He received the 1995 F. E. Terman Award of the American Society for Engineering Education, sponsored by Hewlett Packard Co., for his contributions to engineering education, especially the book Multirate systems and filter banks published by Prentice Hall in 1993. He has given several plenary talks including at the Sampta'01, Eusipco'98, SPCOM'95, and Asilomar'88 conferences on signal processing. He has been chosen a distinguished lecturer for the IEEE Signal Processing Society for the year 1996-97. In 1999 he was chosen to receive the IEEE CAS Society's Golden Jubilee Medal. He is a recepient of the IEEE Signal Processing Society's Technical Achievement Award for the year 2002.

List of Figures

List of Tables