OPTIMAL ALIGNMENT ALGORITHM FOR CONTEXT-SENSITIVE HIDDEN MARKOV MODELS

Byung-Jun Yoon and P. P. Vaidyanathan

Dept. of Electrical Engineering
California Institute of Technology, Pasadena, CA 91125, USA
E-mail: bjyoon@caltech.edu, ppvnath@systems.caltech.edu

ABSTRACT

The hidden Markov model is well-known for its efficiency in modeling short-term dependencies between adjacent samples. However, it cannot be used for modeling longer-range interactions between symbols that are distant from each other. In this paper, we introduce the concept of context-sensitive HMM that is capable of modeling strong pairwise correlations between distant symbols. Based on this model, we propose a polynomial-time algorithm that can be used for finding the optimal state sequence of an observed symbol string. The proposed model is especially useful in modeling palindromes, which has an important application in RNA secondary structure analysis.

1. INTRODUCTION

The hidden Markov model (HMM) has been widely used in many areas due to its efficiency in modeling short-term dependencies between adjacent samples. Traditionally, HMMs have been successfully applied to speech recognition [1], and many speech recognition systems are built upon HMMs and their variants. More recently, HMMs have become also very popular in computational biology. They have been proven to be useful in various problems such as gene identification [2, 3, 4], multiple sequence alignment [4, 5], and so forth.

Despite its many advantages, the HMM has also some limitations. For example, even though it can efficiently model sequences with strong correlation between adjacent samples, it cannot represent longer-range interactions between samples that are distant from each other. Therefore, the resulting structure is always linear, or sequential, and more complex structures with non-sequential dependencies cannot be effectively generated.

According to the Chomsky hierarchy of transformational grammars [7], HMMs can be viewed as the stochastic version of the so-called *regular grammars*. There are four classes in the Chomsky hierarchy as shown in Fig. 1. The regular grammars are the simplest among the four, and they have the most restricted structure, or production rules. However, due to these restrictions, they have efficient algorithms such as the Viterbi's algorithm [1] for finding the optimal state sequence, and Baum-Welch algorithm [8] for re-estimation of the model parameters. Other transformational grammars have less restrictions in their production rules, hence they have more descriptional power to represent complex structures.

Work supported in part by the ONR grant N00014-99-1-1002 and by the NSF grant CCF-0428326, USA.

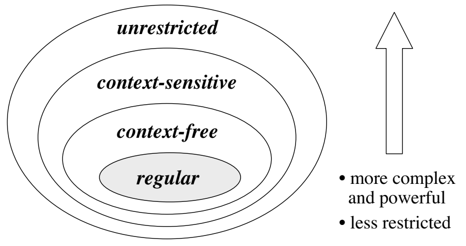

Fig. 1. The Chomsky hierarchy of transformational grammars nested according to the restrictions on their production rules.

However, the computational complexity for parsing the observation sequence increases very quickly, which sometimes makes the use of higher level grammars impractical.

Two classic examples of languages that cannot be modeled using the regular grammars - or equivalently, using HMMs - are the palindrome language and the copy language [7]. The palindrome language includes all sequences that read the same forwards and backwards. The copy language includes all sequences that consists of the concatenation of two identical sequences. Examples of these languages are shown in Fig. 2. The lines in Fig. 2 that connect two symbols indicate the correlation between those symbols which are located distantly from each other. This kind of longer-range interactions between symbols cannot be modeled using regular grammars. It is of course possible that a regular grammar generates such a palindromic sequence as part of its language. However, it is not capable of generating *only such palindromes*, thus not able to effectively differentiate palindromic sequences from non-palindromic ones. In order to simulate such languages, we have to use higher level grammars than the regular grammars in the Chomsky hierarchy. For example, palindromes can be generated by context-free grammars, and we need context-sensitive grammars to represent the copy language.

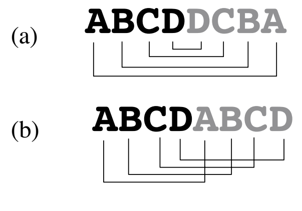

Fig. 2. Examples of (a) palindrome language and (b) copy language. The lines show the correlations between distant symbols.

In this paper, we introduce the concept of context-sensitive HMMs that are capable of modeling longer-range correlations between distant symbols. This model equips some of the states with auxiliary memory elements that store part of the past output symbols, which serve as the *context* of the system. This context affects the emission probabilities as well as the transition probabilities of the HMM, which enables the model generate palindromes, repetitive sequences, and so forth.

One important application of context-sensitive HMMs is the RNA secondary structure analysis. It is known that many interesting RNAs conserve their secondary structures as well as their primary sequences [4, 6]. These RNAs display complex interactions between bases that are distant from each other, and many of them resemble palindromes. The proposed model can be applied for identifying these secondary structures.

2. CONTEXT-SENSITIVE HMM

The proposed model consists of three kinds of states, namely, S_n, P_n and C_n. S_n are *single-emission states*, which are exactly the same as the states used in traditional HMMs. P_n are *pairwise-emission states* and C_n are *context-sensitive states*. These two states always exist in pairs. Each (P_n, C_n) pair has an auxiliary memory dedicated to it, such as a stack or a queue. These two states access the same memory element, and the data stored in this memory serves as the *context* of the system, which decides the transition probabilities and the emission probabilities of the HMM.

The differences between these three states are as follows. In a single-emission state, a symbol is emitted according to the emission probabilities of the state S_n. After the emission, a transition is made from S_n to another state, following the transition probabilities associated with S_n. In a pairwise-emission state, everything is the same as in single-emission states, except that the emitted symbol is stored in the associated memory element before making a transition to the next state. Now, let us consider the context-sensitive states. In a context-sensitive state C_n, the emission probabilities and the transition probabilities are affected by the data that is stored in the associated memory. When entering C_n, this memory element is accessed and one symbol is pulled out. Let us denote this symbol as x. Once the symbol x is retrieved, the emission probabilities are adjusted according to x. For example, the emission probabilities may be adjusted such that the state emits the same symbol x with high probability (possibly, with probability one). Another peculiarity of the context-sensitive state C_n is the fact that the associated memory element is examined before any state attempts to make a transition to it. If the memory is empty, transition to C_n is not allowed, and it is forced to make a transition to another state. This is done by setting the transition probability to C_n to zero, and adjusting the remaining probabilities correspondingly. This is necessary to maintain the number of P_n states the same as that of C_n states.

Based on this model, we can easily construct HMMs that generate only palindromic sequences. One possible example is shown in Fig. 3. In this model, the pair (P_1, C_1) is associated with a stack. The model begins at the pairwise-emission state P_1. The state emits symbols according to its emission probabilities, and the emitted symbols are pushed onto the stack. After making several self-transitions, it finally moves to the context-sensitive state C_1. We adjust the emission probabilities and the transition probabilities of C_1 such that it always emits the symbol on the top of the stack and makes self-transitions while there are symbols left in the

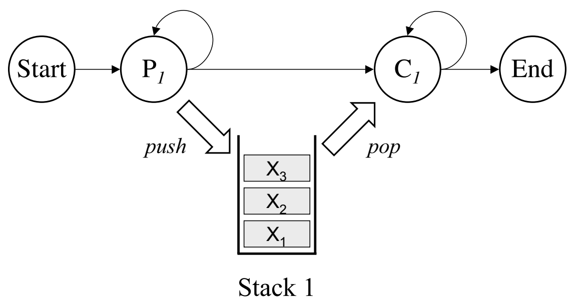

Fig. 3. An example of a context-sensitive HMM that generates palindromes.

stack. In this way, C_1 will emit the same symbols as were emitted by P_1, but in the reverse order. Therefore, the generated string will be always a palindrome. Similarly, we can replace the stack in Fig. 3 by a queue[1] to simulate a copy language. In this case, C_1 will emit the same symbols as the ones emitted by P_1, but this time, in the same order.

3. FINDING THE MOST PROBABLE PATH

Now, let us focus on the case when all the memory elements that are associated with the (P_n, C_n) pairs are stacks. Moreover, we will restrict the pairwise interactions to be nested as shown in Fig. 2 (a). Therefore, crossing interactions that are shown in Fig. 2 (b) will not be allowed. One important question in HMMs is the following. Given an observed symbol string, what is the optimal state sequence, or *path*, that is most probable? If there are M states and the length of the sequence is L, we have M^L different paths. Of course, this includes also many infeasible paths, and the actual number of all feasible paths depends on the particular structure of the 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, as the sequence becomes longer, this becomes quickly infeasible, since the number of paths increases exponentially with L. In regular HMMs, this problem can be solved efficiently by using the Viterbi's algorithm [1]. Let us consider a sequence of symbols $x_1 x_2 \cdots x_L$ and denote the underlying state of x_i as s_i. The Viterbi's algorithm exploits the fact that if $s_1 \cdots s_{n-1} s_n$ is the optimal path for $x_1 \cdots x_{n-1} x_n$ among all paths that end with the state s_n, then $s_1 \cdots s_{n-1}$ must be the optimal path for $x_1 \cdots x_{n-1}$ among all paths that end with the state s_{n-1}. Therefore, in order to find the optimal path for $x_1 \cdots x_n$ with $s_n = m$, we only have to consider the M optimal paths for $x_1 \cdots x_{n-1}$ that end with $s_{n-1} = 1, \ldots, M$, the transition probability from each of these states to the state $s_n = m$, and the probability of emitting the symbol x_n at the state s_n. This makes the computational complexity of the Viterbi's algorithm only $O(LM^2)$, which is much better than $O(M^L)$.

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

[1]Note that a queue is a *first-in-first-out (FIFO)* system, whereas a stack is a *last-in-first-out (LIFO)* system.

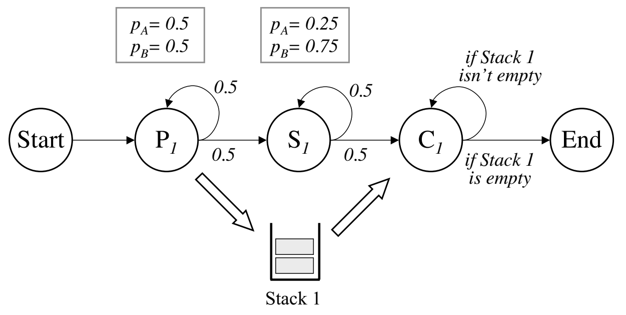

Fig. 4. An example of a simple context-sensitive HMM. This shows that the traditional Viterbi's algorithm cannot be used for finding the most probable path in context-sensitive HMMs.

the probability of a certain path. Therefore, the optimal path for $x_1 \cdots x_n$ cannot be found simply by considering the optimal paths for $x_1 \cdots x_{n-1}$ and extending it.

In order to see this, let us consider the example shown in Fig. 4. This context-sensitive HMM has three states P_1, C_1 and S_1. The emission probabilities and the transition probabilities of P_1 and S_1 are shown in the figure. At the context-sensitive state C_1, a symbol is popped from the stack and emitted. After the emission, the stack is examined to check whether it is empty. If it is empty, the system stops. Otherwise, it makes a self-transition to C_1 and continues. Now, let us consider the sequence $ABBBA$. Assuming that this string comes from the model in Fig. 4, what is the most probable path π^*? It is not difficult to see that there are two feasible paths: $\pi_1 = P_1S_1S_1S_1C_1$ and $\pi_2 = P_1P_1S_1C_1C_1$. Since both paths pass the state S_1 in the middle, let us first consider the optimal path for the first three symbols ABB. Let us denote the sub-paths of π_1 and π_2 up to the third symbol as $\hat{\pi}_1 = P_1S_1S_1$ and $\hat{\pi}_2 = P_1P_1S_1$, respectively. If we compute the probabilities of $\hat{\pi}_1$ and $\hat{\pi}_2$, we get

$$P(\hat{\pi}_1) = \frac{9}{128}, \; P(\hat{\pi}_2) = \frac{6}{128}, \tag{1}$$

hence the optimal path for the first three symbols ABB is $\hat{\pi}_1$. However, if we compute the probabilities of the two paths π_1 and π_2, we obtain

$$P(\pi_1) = \frac{27}{2048}, \; P(\pi_2) = \frac{48}{2048}, \tag{2}$$

which shows that the optimal path for $ABBBA$ is π_2. Apparently, the globally optimal path $\pi^* = \pi_2$ is not an extension of $\hat{\pi}_1$, and this clearly demonstrates that the Viterbi's algorithm cannot be used for finding the most probable path in context-sensitive HMMs.

4. POLYNOMIAL TIME ALGORITHM FOR OPTIMAL ALIGNMENT

Since the number of paths increases exponentially with the length of the sequence, without an efficient algorithm for finding the most probable path, the context-sensitive HMM may not be of much practical value. In this section, we introduce a systematic algorithm for finding the optimal alignment between the observed symbol string and the given model. The proposed algorithm is conceptually similar to the Cocke-Younger-Kasami algorithm [9] that is used for parsing stochastic context-free grammars (SCFGs). It will be shown later that the computational complexity of the proposed algorithm is lower than that of the CYK algorithm.

Let us first define the variables that are needed in the algorithm. Let $\mathbf{x} = x_1x_2...x_L$ be the observed symbol string, where L is the length of the string. We assume that there are M states in the context-sensitive HMM, which we simply denote by $1, 2, \ldots, M$. We assume that there are N pairs (P_n, C_n) of pairwise-emission states and context-sensitive states, where a separate stack is dedicated to each. These $2N$ states are included in the set of M states $\{1, \ldots, M\}$. It is assumed that all pairwise interactions between P_n and C_n are nested and do not cross each other. Let us also define the sets $\mathcal{P} = \{P_1, \ldots, P_N\}$, $\mathcal{C} = \{C_1, \ldots, C_N\}$ and $\mathcal{R} = \{(P_1, C_1), \ldots, (P_N, C_N)\}$ for notational convenience. We denote the transition probability from state v to w as $t(v, w)$, and the emission probability of a symbol x at a state v as $e(x|v)$. Now, let us define the variable $\gamma(i, j, v, w)$ which is the log-probability of the optimal path among all paths $s_i \cdots s_j$ with $s_i = v$ and $s_j = w$, where all pairwise-emission states P_n are paired with the corresponding context-sensitive state C_n inside the path. This will ultimately lead to the log-probability $\log P(\mathbf{x}, \pi^*|\Theta)$ of the optimal path π^*, where Θ is the set of model parameters. Finally, we define the variable $\lambda(i, j, v, w)$ that will be used for tracing back the optimal path.

4.1. Computing the log-probability of the optimal path

The optimal alignment algorithm is defined as follows.

Initialization

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

$$\begin{aligned} \gamma(i,i,v,v) &= \begin{cases} \log e(x_i|v) & v \notin \mathcal{P}, \mathcal{C} \\ -\infty & \text{otherwise} \end{cases} \\ \lambda(i,i,v,v) &= (0,0,0,0) \end{aligned}$$

Iteration

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

(i) $v = P_n, w = C_m (n \neq m)$, or $v \in \mathcal{C}$ or $w \in \mathcal{P}$

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

(ii) $(v, w) \in \mathcal{R}, j = i + 1$

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

(iii) $(v, w) \in \mathcal{R}, j \neq i + 1$

$$\begin{aligned} \gamma(i,j,v,w) &= \max_{u_1,u_2} \Big[\log e(x_i|v) + \log t(v,u_1) \\ &\quad + \gamma(i+1,j-1,u_1,u_2) + \log t(u_2,w) + \log e(x_j|w) \Big] \\ (u_1^*, u_2^*) &= \operatorname*{arg\,max}_{(u_1,u_2)} \Big[\log e(x_i|v) + \log t(v,u_1) \\ &\quad + \gamma(i+1,j-1,u_1,u_2) + \log t(u_2,w) + \log e(x_j|w) \Big] \\ \lambda(i,j,v,w) &= (i+1, j-1, u_1^*, u_2^*) \end{aligned}$$

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

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

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

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

(vi) $v \notin \mathcal{P}, w \notin \mathcal{C}$

In this case, the variables $\gamma(i,j,v,w)$ and $\lambda(i,j,v,w)$ can be updated using any of the update formulae in (iii)$\sim$(v).

Termination

$$\begin{aligned}
\log P(\mathbf{x},\pi^*|\Theta) &= \max_{v,w} \Big[\log t(0,v) + \gamma(1,L,v,w) \\
&\qquad + \log t(w,0)\Big] \\
(v^*,w^*) &= \arg\max_{(v,w)} \Big[\log t(0,v) + \gamma(1,L,v,w) \\
&\qquad + \log t(w,0)\Big] \\
\lambda^* &= (1,L,v^*,w^*) \qquad \square
\end{aligned}$$

The proposed algorithm starts from the inside of the observation sequence, and proceeds to the outward direction, to find the optimal path iteratively. It should be noted that every-time there is an interaction between s_i and s_j, they are considered at the same time as shown in (ii) and (iii) of the *iteration* step. This informs us of the symbol x_i that was emitted by P_n, hence we can adjust the probabilities of the corresponding state C_n according to this value.

4.2. Trace-back

Let us define $\lambda_k = (i_k, j_k, v_k, w_k)$. We also need two stacks T_ℓ, T_r for trace-back. The optimal path is traced back as follows.

Initialization

Let $k = 0$. Initialize $\lambda_0 = \lambda^* = (1, L, v^*, w^*)$.

Iteration

```
$\lambda_{k+1} \Longleftarrow \lambda(\lambda_k) = \lambda(i_k, j_k, v_k, w_k)$
If $i_{k+1} \neq i_k$ then push $v_k$ onto stack $T_\ell$.
If $j_{k+1} \neq j_k$ then push $w_k$ onto stack $T_r$.
If $\lambda_{k+1} = (0,0,0,0)$ goto the termination step.
Otherwise, increment $k$ and repeat the iteration step.
```

Termination

```
Pop $v$ from $T_\ell$ and push it onto $T_r$ until $T_\ell$ becomes empty.
Initialize $\pi^* = \phi$ (null string).
Pop $v$ from $T_r$ and append it to the right of $\pi^*$.
Repeat until $T_r$ becomes empty.                    $\square$
```

At the end of this procedure, we get the most probable path π^*. It is not difficult to see that the computational complexity of the alignment algorithm is $O(L^2M^3)$, which is much better than $O(M^L)$ of the exhaustive search, and also smaller than the complexity $O(L^3M^3)$ of the CYK algorithm for general SCFGs.

5. CONCLUDING REMARKS

In this paper, we proposed the concept of context-sensitive HMMs that can be used for modeling pairwise interactions between distant symbols. The proposed model has a greater descriptive power than the traditional HMMs. For example, it can be used for modeling palindromic languages, copy languages, and so forth. The proposed model can be used for modeling RNA secondary structures, since many interesting RNAs look like palindromes [4, 6]. SCFGs have been used for modeling RNAs [4, 6], and we may use context-sensitive HMMs instead [10], instead of SCFGs. This is a topic for future research.

6. REFERENCES

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

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

[3] S. L. Salzberg, A. L. Delcher, S. Kasif, O. White, "Microbial gene identification using interpolated Markov models", Nucleic Acids Res. 26 (2) (1998) 544-548.

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

[5] S. R. Eddy, "Multiple alignment using hidden Markov models", Proceedings of the Third International Converence on Intelligent Systems for Molecular Biology (1995) 112-120.

[6] S. R. Eddy, "Computational genomics of noncoding RNA genes", Cell 109 (2) (2002) 137-40.

[7] N. Chomsky, "On certain formal properties of grammars", Information and Control (2) (1959) 137-167.

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

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

[10] Byung-Jun Yoon and P. P. Vaidyanathan, "HMM with auxiliary memory: a new tool for modeling RNA secondary structures", Proc. 28th Asilomar Conference on Signals, Systems, and Computers, Monterey, CA, Nov. 2004.