ECE 8540
Analysis of Tracking Systems

Assignment 7
Particle Filter

Vivek Koodli Udupa
C12768888

November 29, 2018

1 Introduction

This report considers the problem of modeling a system described by Hidden Markov Model(HMM). A Markov model is a stochastic model used to model randomly changing systems. A Markov process is a random process where the future state is independent of the past states given the present state. A Hidden Markov Model is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (i.e. hidden) states. The aim is to discover the hidden state sequence that most likely describes a given observation sequence.

One solution to this problem is to use the Viterbi algorithm, which finds the single best state sequence for an observation sequence. For example, in speech-to-text (speech recognition), the acoustic signal is treated as the observed sequence of events, and a string of text is considered to be the "hidden cause" of the acoustic signal. The Viterbi algorithm finds the most likely string of text given the acoustic signal.

The problem considered for this report is a simple HMM which comprises of two states, H(High GC content) and L(Low GC content). State H characterizes coding DNA while state L characterizes non-coding DNA. The model is used to predict the region of coding DNA from two given sequences 'GGCACTGAA' and 'TCAGCGGCT'.

2 Methods

This section describes the implementation of the Viterbi algorithm to find the hidden state probabilities for the given problem.

2.1 Description of the HMM

The HMM in consideration is shown in Figure 1. The model consists of two states, labeled H and L in the example, which can be given numerical values of 0 and 1. The prior probabilities are 0.5, 0.5. The state transition probabilities are 0.5, 0.5 for state 0 and 0.4, 0.6 for state 1. Each state observes a discrete value that takes on one of four values A, C, G, T that can be given numerical values 0, 1, 2, 3. The emission probabilities of these values are 0.2, 0.3, 0.3, 0.2 for state 0 and 0.3, 0.2, 0.2, 0.3 for state 1.

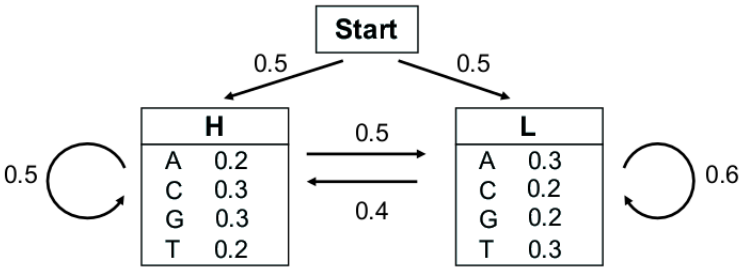

Figure 1: The Hidden Markov Model with H and L States

The Viterbi calculations are done using sums of log_2 probabilities.

2.2 Implementation of Viterbi algorithm

There are several paths that lead to the desired sequence of states, but not all of them have the same probabilities. The Viterbi algorithm is a dynamical programming algorithm that computes the most probable path.

Consider a HMM with k states. The probability of the most probable path ending in state l with observation i is:

$$p_l(i,x) = e_l(i) \cdot \max_k(p_k(j,x-1) \cdot p_{kl}) \tag{1}$$

where $e_l(i)$ is the probability to observe element i in state l. $p_k(j,x-1)$ is the probability of the most probable path ending at position x-1 in state k with element j. p_{kl} is probability of the transition from state l to state k.

For the given sequence "GGCACTGAA", the probability of the most probable path ending in state 'H' with observation 'A' at the fourth position (given C at third position) is calculated as:

$$p_H(A,4) = e_H(A) \cdot max(p_L(C,3) \cdot p_{LH}, p_H(C,3) \cdot p_{HH}) \tag{2}$$

Similarly the most probably path can be computed recursively from the first to the last element in the given sequence.

2.3 Backtracking

Once the probabilities for all possible states are calculated, the path to the maximum probability is backtracked to find the best path. The state with the highest probability is chosen. In cases where both the states have the same probability value, same path as the previous state is maintained.

3 Results

This sections consists of the results i.e. the table of max probabilities obtained from the Viterbi algorithm for the two given sequences. The calculation was done using log_2 of probabilities.

3.1 Input Sequence 1

Input Sequence	G	G	C	A	C	T	G	A	A
State H	**-2.73**	**-5.47**	**-8.21**	-11.53	-14.00	-17.32	-19.53	-22.86	-25.65
State L	-3.32	-6.05	-8.79	**-10.94**	**-14.00**	**-16.48**	**-19.53**	**-22.01**	**-24.48**
Best Path	H	H	H	L	L	L	L	L	L

Table 1: Table of maximum probabilities for sequence "GGCACTGAA"

Table 1 shows the best probable path for the given sequence "GGCACTGAA" is HHHLLLLLL and the maximum probability is $2^{-24.48} = 4.2517 \times 10^{-08}$. The boldfaced numbers represent the highest probability for that particular event.

3.2 Input Sequence 2

Input Sequence	T	C	A	G	C	G	G	C	T
State H	-3.32	-5.79	-9.11	**-11.32**	**-14.06**	**-16.80**	**-19.53**	**-22.27**	-25.59
State L	**-2.73**	**-5.79**	**-8.26**	-11.32	-14.38	-17.38	-20.12	-22.86	**-25.01**
Best Path	L	L	L	H	H	H	H	H	L

Table 2: Table of maximum probabilities for sequence "TCAGCGGCT"

Table 2 shows the best probable path for the given sequence "TCAGCGGCT" is LLLHHHHHL and the maximum probability is $2^{-25.01} = 2.9525 \times 10^{-08}$. The boldfaced numbers represent the highest probability for that particular event.

4 Conclusion

The Viterbi algorithm is used to compute the most probable path (as well as its probability). It requires knowledge of the parameters of the HMM model and a particular output sequence and it finds the state sequence that is most likely to have generated that output sequence. It works by finding a maximum over all possible state sequences.

There are often many state sequences that can produce the same particular output sequence, but with different probabilities. It is possible to calculate the probability for the HMM model to generate that output sequence by doing the summation over all possible state sequences.

References

[1] https://en.wikipedia.org/wiki/Viterbi_algorithm

[2] https://en.wikipedia.org/wiki/Hidden_Markov_model

[3] http://cecas.clemson.edu/~ahoover/ece854/

Appendix

Main

```
function [Prob, Best] = run()%Main function to execute the Viterbi algorithm
clc
clear
    %% Initialization
    %Order      A     C     G      T
    H = log2( [0.2, 0.3, 0.3, 0.2] );
    L = log2( [0.3, 0.2, 0.2, 0.3] );


    %Initial probability
    poh = -1;
    pol = -1;

    %Transition probabilities
    phh = -1;
    pll = log2(0.6);
    phl = -1;
    plh = log2(0.4);
```

```
    %Input sequence

%       IS = ['G', 'G', 'C', 'A', 'C', 'T', 'G', 'A', 'A' ]; %Sequence 1
    IS = ['T', 'C', 'A', 'G', 'C', 'G', 'G', 'C', 'T' ]; %Sequence 2

    size = length(IS);

    ISn = zeros(1,size);

    %Probabilities row1 -> Ph ;row2 -> Pl ;row3 -> best prob
    Prob = zeros(4, size);
    Best = zeros(1, size);

    %Convert sequence to array
    ISn = Convert(IS, ISn);

    %% Execution

    %Calculating the forward Probabilities using Viterbi Algorithm
    Prob = Viterbi(Prob, poh, pol, plh, pll, phh, phl, H, L, ISn);

    %Backtracking
    Best = Backtracking(Best, Prob);

    disp("Best Probs: ");
    disp(Best)

    for i = 1:size
        if(Best(i) == 1)
            Ans(i) = 'L';
        else
            Ans(i) = 'H';
        end
    end

    disp(Ans);



disp("SUCCESS!!")
end
```

Viterbi algorithm implementation

```
function Prob = Viterbi(Prob, poh, pol, plh, pll, phh, phl, H, L, Input)
```

```
%Viterbi algorithm to calculate the forward probabilities
    size = length(Input);

    for i = 1:size
        if( i == 1)
            Prob(1,i) = poh + H(Input(i));
            Prob(2,i) = pol + L(Input(i));

            if( Prob(1,i) == Prob(2,i) )
                Prob(3,i) = Prob(1,i);
                Prob(4,i) = 9; %Equal

            elseif(Prob(1,i) > Prob(2,i))
                Prob(4,i) = 0;
                Prob(3,i) = Prob(1,i); %High

            else
                Prob(4,i) = 1; %Low
                Prob(3,i) = Prob(2,i);


            end

        else
            Prob(1,i) = H(Input(i)) + max ( (Prob(1, i-1) + phh), (Prob(2, i-1)
            Prob(2,i) = L(Input(i)) + max ( (Prob(1, i-1) + phl), (Prob(2, i-1)

            if( (Prob(1,i) - Prob(2,i)) == 0 )
                Prob(3,i) = Prob(1,i); %Same
                Prob(4,i) = 9;

            elseif(  Prob(1,i) < Prob(2,i))
                Prob(3,i) = Prob(2,i);
                Prob(4,i) = 1; %Low

            else
                Prob(3,i) = Prob(1,i);
                Prob(4,i) = 0; %High

            end


        end
    end
```

```
end
```

Backtracking

```
function Best = Backtracking(Best, Prob)
%Finds the best probsbility

    init = 1;
    finalProb = Prob(4,:);
    size = length(finalProb);

    for i = size:-1:1
        if(finalProb(i) == 9)
            if(i == size)
                Best(i) = init; %If last prob is same, initialize
            else
                Best(i) = Best(i+1);
            end
        elseif(finalProb(i) == 0)
            Best(i) = 0; %Low
        else
            Best(i) = 1; %High
        end
    end

end
```

Conversion

```
function Num = Convert(Alpha, Num)
%Convert alphabetical sequence to numbers
    for i = 1:length(Alpha)
       if(Alpha(i) == 'A')
           Num(i) = 1;
       elseif(Alpha(i) == 'C')
           Num(i) = 2;
       elseif(Alpha(i) == 'G')
           Num(i) = 3;
       else
           Num(i) = 4;
       end
    end

end
```