I529: Machine Learning in Bioinformatics (Spring 2018)

Hidden Markov Models

Yuzhen Ye
School of Informatics and Computing
Indiana University, Bloomington
Spring 2018

Outline

- Review of Markov chain & CpG island
- HMM: three questions & three algorithms
  - Q1: most probable state path—Viterbi algorithm
  - Q2: probability of a sequence p(x)—Forward algorithm
  - Q3: Posterior decoding (the distribution of S_i, given x)—Forward/backward algorithm
- Applications
  - CpG island (problem 2)
  - Eukaryotic gene finding (Genscan)
  - Gene prediction for metagenomics (FragGeneScan)
- Generalized HMM (GHMM)
  - A state emits a *string* according to a probability distribution
  - Viterbi decoding for GHMM

1st order Markov chain

An ***integer time stochastic process***, consisting of a set of ***m*>1** states $\{s_1,\ldots,s_m\}$ and

1. An ***m*** dimensional ***initial distribution vector*** $(p(s_1),.., p(s_m))$.
2. An ***m ×m transition probabilities matrix M***= $(a_{s_i s_j})$

For example, for DNA sequence, the states are $\{A, C, T, G\}$, $p(A)$ the probability of A to be the 1st letter in a DNA sequence, and a_{AG} the probability that G follows A in a sequence.

Example: *CpG* Island

- We consider two questions (and some variants):
  - **Question 1:** Given a short stretch of genomic data, does it come from a CpG island ?
  - **Question 2:** Given a long piece of genomic data, does it contain CpG islands in it, where, and how long?
- We "solve" the first question by modeling sequences with and without CpG islands as Markov Chains over the same states {A,C,G,T} but different transition probabilities.

Question 2: Finding CpG Islands

Given a long genomic string with possible CpG Islands, we define a Markov Chain over 8 states, all interconnected:

A^+	C^+	G^+	T^+
A^-	C^-	G^-	T^-

The problem is that we don't know the sequence of ***states*** which are traversed, but just the sequence of ***letters***.

Therefore we use here ***Hidden Markov Model***

The fair bet casino problem

- The game is to flip coins, which results in only two possible outcomes: **H**ead or **T**ail.
- The **F**air coin will give **H**eads and **T**ails with same probability ½.
- The **B**iased coin will give **H**eads with prob. ¾.
- Thus, we define the probabilities:
  - P(H|F) = P(T|F) = ½
  - P(H|B) = ¾, P(T|B) = ¼
  - The crooked dealer changes between Fair and Biased coins with probability 0.1

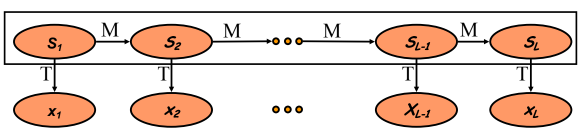
The fair bet casino problem

- **Input:** A sequence $x = x_1x_2x_3\ldots x_n$ of coin tosses made by two possible coins (***F*** or ***B***).
- **Output:** A sequence $S = s_1\ s_2\ s_3\ldots\ s_n$, with each s_i being either F or B indicating that x_i is the result of tossing the Fair or Biased coin, respectively.

Hidden Markov model (HMM)

- Can be viewed as an abstract machine with *k* *hidden* states that emits symbols from an alphabet Σ.
- Each state has its own probability distribution, and the machine switches between states according to this probability distribution.
- While in a certain state, the machine makes 2 decisions:
  - What state should I move to next?
  - What symbol - from the alphabet Σ - should I emit?

Parameters defining a HMM



HMM consists of:

A Markov chain over a set of (hidden) states, and for each state ***s*** and observable symbol ***x***, an emission probability $\boldsymbol{p}(\boldsymbol{X_i}=\boldsymbol{x}|\boldsymbol{S_i}=\boldsymbol{s})$.

<u>A set of parameters defining a HMM:</u>

Markov chain initial probabilities: $p(S_1 = t) = b_t \rightarrow p(s_1|s_0)=p(s_1)$

Markov chain transition probabilities: $p(S_{i+1} = t|S_i = s) = a_{st}$

Emission probabilities: $\boldsymbol{p}(X_i = b|\ S_i = s) = e_s(b)$

Why "hidden"?

- Observers can see the emitted symbols of an HMM but have *no ability to know which state the HMM is currently in*.
- Thus, the goal is to infer the most likely hidden states of an HMM based on the given sequence of emitted symbols.

Example: *CpG* island

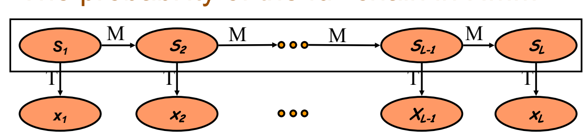
Question 2: Given a long piece of genomic data, does it contain CpG islands in it, where, and how long?

Hidden Markov Model: this seems a straightforward model (but we will discuss later why this model is NOT good).

Hidden states: { '+' , '-' }

Observable symbols: {A, C, G, T}

The probability of the full chain in HMM



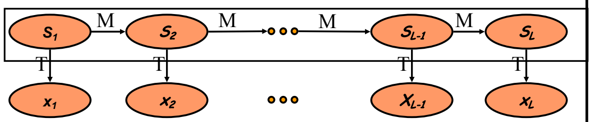
For the full chain in HMM we assume the probability:

$$p(S,X) = p(s_1 \cdots s_L, x_1 \cdots x_L) = \prod_{i=1}^{L} p(s_i|s_{i-1})p(x_i|s_i) = \prod_{i=1}^{L} a_{s_{i-1},s_i} e_{s_i}(x_i)$$

The probability distribution over all sequences of length *L*,

$$\sum_{(s_1,\ldots,s_L;x_1,\ldots,x_L)} p(s_1,\ldots,s_L;x_1,\ldots,x_L) = \sum_{(s_1,\ldots,s_L;x_1,\ldots,x_L)} \left[\prod_{i=1}^{L} a_{s_{i-1},s_i} e_{s_i}(x_i)\right] = 1$$

Three common questions



3 questions of interest, given a HMM:

Given the "visible" observation sequence $x=(x_1,\ldots,x_L)$, find:

1. A ***most probable*** (hidden) ***path***
2. The probability of x
3. For each $i = 1,..,L$, and for each state k, the probability that $s_i=k$.

Q1. Most probable state path

Given an output sequence $x = (x_1,\ldots,x_L)$,

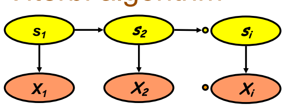
A ***most probable*** path $s^*= (s^*_1,\ldots,s^*_L)$ is one which maximizes $p(s|x)$.

$$s^* = (s_1^*,\ldots,s_L^*) = \operatorname*{argmax}_{(s_1,\ldots,s_L)} p(s_1,\ldots,s_L | x_1,\ldots,x_L)$$

Since $$p(S|X) = \frac{p(S,X)}{p(X)} \;\alpha\; p(S,X)$$

we need to find s which maximizes $p(s,x)$

Viterbi algorithm



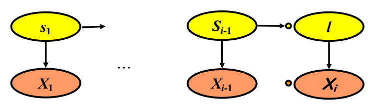
The task: compute $\operatorname*{argmax}_{(s_1,\ldots,s_L)} p(s_1,\ldots,s_L; x_1,\ldots,x_L)$

Let the states be $\{1,\ldots,m\}$

Idea: for $i=1,\ldots,L$ and for each state l, compute:

$v_l(i)$ = the probability $p(s_1,..,s_i;x_1,..,x_i|s_i=l)$ of the most probable path up to i, which ends in state l.

Viterbi algorithm

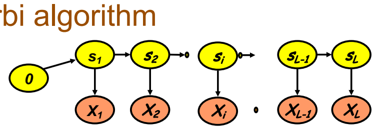


$v_l(i)$ = the probability $p(s_1,..,s_i;x_1,..,x_i|s_i=l)$ of the most probable path up to i, which ends in state l.

For $i = 1,\ldots,L$ and for each state l we have:

$$v_l(i) = e_l(x_i) \cdot \max_k \{v_k(i-1) \cdot a_{kl}\}$$

Viterbi algorithm



Add the special initial state 0

Initialization: $v_0(0) = 1$, $v_k(0) = 0$ for $k > 0$

For i=1 to L do for each state l :

$v_l(i) = e_l(x_i)\ \max_k \{v_k(i\text{-}1)a_{kl}\}$

$ptr_i(l)=argmax_k\{v_k(i\text{-}1)a_{kl}\}$

//storing previous state for retrieving the path

Termination: $s_L^*=\max_k\{v_k(L)\}$

Result: $p(s_1^*,\ldots,s_L^*;x_1,\ldots,x_l)$, where $s_i^*=ptl_{i+1}(s_{i+1}^*)$

Example: a fair casino problem

HMM: hidden states {F(air), L(oaded)}, observation symbols {H(ead), T(ail)}

Transition probabilities

	F	L
F	0.9	0.1
L	0.1	0.9

Emission probabilities

	H	T
F	1/2	1/2
L	3/4	1/4

Initial prob.

P(F)=P(L)=1/2

Find the most likely state sequence for the observation sequence: HHTH

Q2. Computing $p(x)$

Given an output sequence $\boldsymbol{x} = (x_1,\ldots,x_L)$, compute the probability that this sequence was generated by the given HMM:

$$p(X) = \sum_S p(X,S)$$

The summation taken over all state-paths $\boldsymbol{s}$ generating $\boldsymbol{x}$.

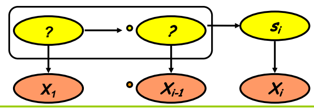
Forward algorithm

The task: compute $p(x) = \sum_s p(x,s)$

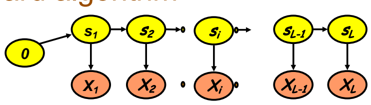
Idea: for i=1,…,L and for each state l, compute:

$f_l(i) = p(x_1,\ldots,x_i;s_i=l)$, the probability of ***all*** the paths which emit $(x_1,..,x_i)$ and end in state $s_i=l$.

Recursive formula: $f_l(i) = e_l(x_i)\sum_k f_k(i-1)a_{kl}$



Forward algorithm



Similar to the Viterbi algorithm (use **sum instead of maximum**):

Initialization: $f_0(0) := 1$, $f_k(0) := 0$ for $k>0$

For i=1 to L do for each state l :

$$f_l(i) = e_l(x_i)\sum_k f_k(i-1)a_{kl}$$

Result: $p(x_1,\ldots,x_L) = \sum_k f_k(L)$

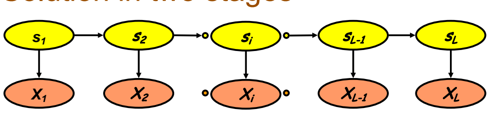
Q3. Distribution of S_i, given x

Given an output sequence $\boldsymbol{x} = (x_1,\ldots,x_L)$,

Compute for each i=1,…,l and for each state k the probability that $s_i = k$.

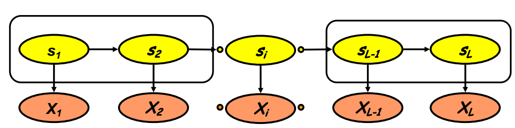
This helps to reply queries like: what is the probability that s_i is in a CpG island, etc.

Solution in two stages



1. For a fixed i and each state k, an algorithm to compute $p(s_i=k \mid x_1,\ldots,x_L)$.
2. An algorithm which performs this task **for every i = 1,..,L**, without repeating the first task L times.

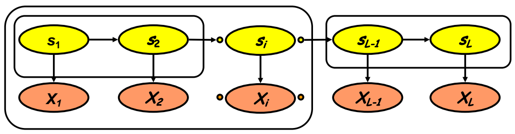
Computing for a single i:



$$p(s_i \mid x_1,\ldots,x_L) = \frac{p(s_i,x_1,\ldots,x_L)}{p(x_1,\ldots,x_L)}$$

$$\alpha \;\; p(s_i,x_1,\ldots,x_L)$$

Computing for a single *i*:

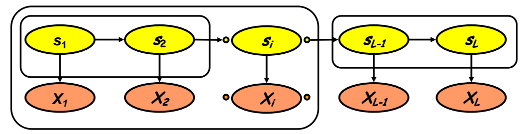


$p(x_1,\ldots,x_L,s_i) = p(x_1,\ldots,x_i,s_i)\, p(x_{i+1},\ldots,x_L \mid x_1,\ldots,x_i,s_i)$

(by the equality $p(A,B) = p(A)p(B|A)$).

$p(x_1,\ldots,x_i,s_i) = f_{s_i}(i) \equiv F(s_i)$, which is computed by the forward algorithm.

$B(s_i)$: The backward algorithm



$p(x_1,\ldots,x_L,s_i) = p(x_1,\ldots,x_i,s_i)\, p(x_{i+1},\ldots,x_L \mid x_1,\ldots,x_i,s_i)$

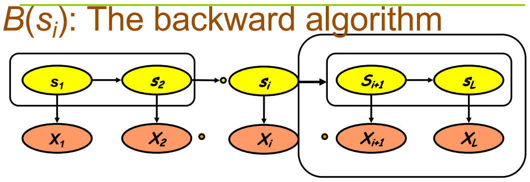
We are left with the task to compute the ***Backward algorithm***

$\boldsymbol{b}(s_i) \equiv p(x_{i+1},\ldots,x_L \mid x_1,\ldots,x_i,s_i)$,

and get the desired result:

$p(x_1,\ldots,x_L,s_i) = p(x_1,\ldots,x_i,s_i)\, p(x_{i+1},\ldots,x_L \mid s_i) \equiv f(s_i)\cdot b(s_i)$

$B(s_i)$: The backward algorithm

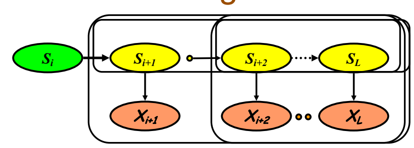


From the probability distribution of Hidden Markov Chain and the definition of conditional probability:

$b(s_i) = p(x_{i+1},\ldots,x_L \mid x_1,\ldots,x_i,s_i) = p(x_{i+1},\ldots,x_L \mid s_i) =$

$$= \sum_{s_{i+1}} a_{s_i,s_{i+1}} e_{s_{i+1}}(x_{i+1}) \underbrace{p(x_{i+2},\ldots,x_L) \mid s_{i+1})}_{b(s_{i+1})}$$

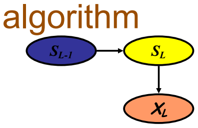
$B(s_i)$: *The* backward algorithm



The Backward algorithm computes $B(s_i)$ from the values of $B(s_{i+1})$ for all states s_{i+1}.

$$b(s_i) = p(x_{i+1} \ldots x_L|s_i) = \textstyle\sum_{s_{i+1}} a_{s_i,s_{i+1}} e_{s_{i+1}}(x_{i+1}) b(s_{i+1})$$

$B(s_i)$: *The* backward algorithm



First step, step L-1:

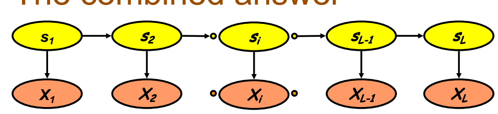
Compute $B(s_{L-1})$ for each possible state s_{L-1}:

$$b(s_{L-1}) = p(x_L \mid s_{L-1}) = \sum_{s_L} a_{s_{L-1},s_L} e_{s_L}(x_L)$$

For i=L-2 down to 1, for each possible state s_i, compute $b(s_i)$ from the values of $b(s_{i+1})$:

$$b(s_i) = p(x_{i+1} \ldots x_L|s_i) = \textstyle\sum_{s_{i+1}} a_{s_i,s_{i+1}} e_{s_{i+1}}(x_{i+1}) b(s_{i+1})$$

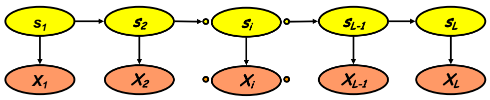
The combined answer



1. To compute the probability that S_i=s_i given $\boldsymbol{x}$=$(x_1,\ldots,x_L)$, run the forward algorithm and compute $f(s_i) = p(x_1,\ldots,x_i,s_i)$, run the backward algorithm to compute $b(s_i) = p(x_{i+1},\ldots,x_L|s_i)$, the product $f(s_i)b(s_i)$ is the answer (for every possible value s_i).

2. To compute these probabilities for every s_i simply run the forward and backward algorithms once, storing $f(s_i)$ and $b(s_i)$ for every i (and every value of s_i). Compute $f(s_i)b(s_i)$ for every i.

Time and space complexity of the viterbi/forward/backward algorithms



Time complexity is $O(m^2L)$ where m is the number of states.
Space complexity is $O(mL)$ (a table).
Both are **linear in the length** of the chain (observation sequence), provided the number of states (m) is a constant.

Example: Finding CpG islands

- Observed symbols: {A, C, G, T}
- Hidden States: { '+' , '-' }
- Transition probabilities:
  - P(+|+), P(-|+), P(+|-), P(-|-)
- Emission probabilities:
  - P(A|+), P(C|+), P(G|+), P(T|+)
  - P(A|-), P(C|-), P(G|-), P(T|-)
- Bad model! – did not model the correlation between adjacent nucleotides!

Example: Finding CpG islands

- Observed symbols: {A, C, G, T}
- Hidden States: {A^+, C^+, G^+, T^+, A^-, C^-, G^-, T^-}
- Emission probabilities:
  - $P(A|A^+)=P(C|C^+)=P(G|G^+)=P(T|T^+)=P(A|A^-)=P(C|C^-)=P(G|G^-)=P(T|T^-)=1.0$; else $P(X|Y)=0$;
- Transition probabilities:
  - 16 probabilities in '+' model; 16 probabilities for '-' model;
  - 16 probabilities for transitions between '+' and '-' models

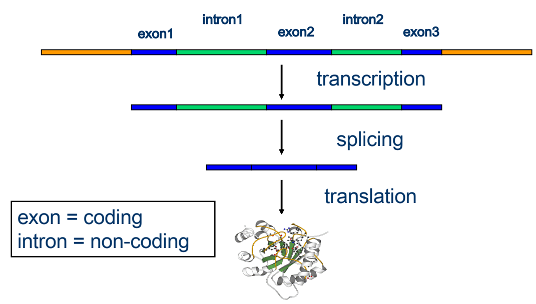
Example: Eukaryotic gene finding

- In eukaryotes, the gene is a combination of coding segments (**exons**) that are interrupted by non-coding segments (**introns**)
- This makes computational gene prediction in eukaryotes even more difficult
- Prokaryotes don't have introns - Genes in prokaryotes are continuous

Example: Eukaryotic gene finding

- On average, vertebrate gene is about 30KB long
- Coding region takes about 1KB
- Exon sizes vary from double digit numbers to kilobases
- An average 5' UTR is about 750 bp
- An average 3'UTR is about 450 bp but both can be much longer.

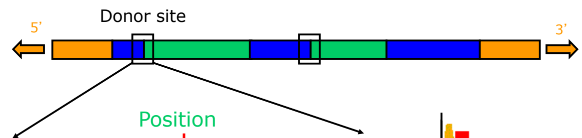
Central dogma and splicing



Splicing signals

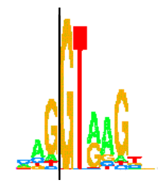
- Try to recognize location of splicing signals at exon-intron junctions
  - This has yielded a weakly conserved donor splice site and acceptor splice site
- Profiles for sites are still weak, and lends the problem to the Hidden Markov Model (HMM) approaches, which capture the statistical dependencies between sites

Modeling splicing signals



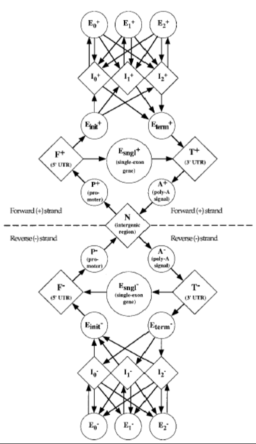
Position

%	-8	…	-2	-1	0	1	2	…	17
A	26	…	60	9	0	1	54	…	21
C	26	…	15	5	0	1	2	…	27
G	25	…	12	78	99	0	41	…	27
T	23	…	13	8	1	98	3	…	25



Genscan model

- Genscan considers the following:
  - Promoter signals
  - Polyadenylation signals
  - Splice signals
  - Probability of coding and non-coding DNA
  - Gene, exon and intron length



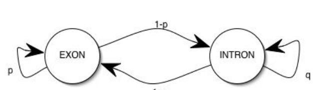
Chris Burge and Samuel Karlin, *Prediction of Complete Gene Structures in Human Genomic DNA*, JMB. (1997) **268**, 78-94

Genscan model

- States correspond to different functional units of a genome (promoter region, intron, exon,….)
- The states for introns and exons are subdivided according to three frames.
- There are two symmetric sub modules for forward and backward strands.
- Performance: 80% exon detecting (but if a gene has more than one exon, the detection accuracy decrease rapidly).

Note: there is no edge pointing from a node to itself in the Markov chain model of Genscan. Why? Because Genscan uses the *Generalized Hidden Markov model* (GHMM), instead of the regular HMM.

State duration



$$P(\text{exon of length } k) = p^k(1-p)$$

Geometric distribution

In the regular HMM, the length distribution of a hidden state (also called the duration) always follow a geometric distribution. In reality, however, the length distribution may be different.

FragGeneScan

- Metagenomic dataset contains sequences from a mixture of species
- Using a general model for prediction of genes in metagenomic sequences/assemblies
- No need to train a model for prediction in each dataset

The effect of sequencing errors on gene prediction

Original gene

QLFAYADT**I**EKQVNNA

CAACTCTTCGCCTACGCCGACACC**A TA**GAAAAACAGGTCAACAACGCCTTAGCCGCG

CAACTCTTCGCCTACGCCGACACC**ACTA**GAAAAACAGGTCAACAACGCCTTAGCCGCG

Read has an sequencing error that cause frame shift

Sequencing errors that cause frame shift can mess up gene prediction (so that gene predictors that rely on ORFs, or partial ORFs may have difficulty dealing with these reads)

43

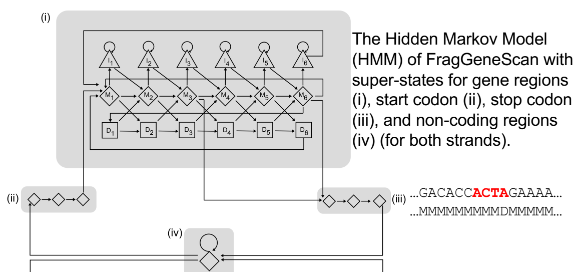
FragGeneScan for gene prediction in short, error-prone reads

- Utilizes a probabilistic model that combines sequencing error models and codon usage to improve the accuracy in predicting protein-coding regions from environmental sequences
- Detects sequencing errors (fixes frameshift)
- *ab initio* predictor (not limited to the availability of the protein databases)

Rho et al, Nucleic Acid Research, 2010

44

FragGeneScan HMM



The Hidden Markov Model (HMM) of FragGeneScan with super-states for gene regions (i), start codon (ii), stop codon (iii), and non-coding regions (iv) (for both strands).

45

Generalized HMMs (Hidden semi-Markov models)

- Based on a variant-length Markov chain;
- The (emitted) output of a state is a string of finite length;
- For a given state, the output string and its length are according to a probability distribution;
- Different states may have different length distributions.

GHMMs

A finite set Σ of hidden states

Initial state probability distribution $b_t=p(s_0)$

Transition probabilities $a_{st}= p(s_i=t|s_{i-1}=s)$ for s, t in Σ; $a_{tt}=0$.

*Length distribution *f* of the states *t* (f_t is the length distribution of state *t*)

*Probabilistic models for each state *t*, according to which output strings are generated upon visiting the state

Segmentation by GHMMs

A **parse** ϕ of an observation sequence $X=(x_1,\ldots x_L)$ of length L is a sequence of hidden states $(s_1,\ldots,s_t)$ with an associated duration d_i to each state s_i, where

$$L = \sum_{i=1}^{t} d_i$$

A parse represents a partition of X, and is equivalent to a hidden state sequence in HMM;

Intuitively, a parse is an annotation of a sequence, matching each segment with a functional unit of a gene

Let $\phi=(s_1,\ldots,s_t)$ be a parse of sequence X;

$P(x_{q+1}x_{q+2}\ldots x_{q+d_i} \mid s_i)$: probability of generating $x_{q+1}x_{q+2}\ldots x_{q+d_i}$ by the sequence generation model of state s_i with length d_i, where $q=\sum_{j=1}^{i-1} d_j$

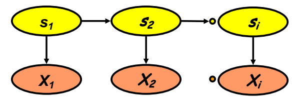
The probability of generating X based on ϕ is

$$P(x_1,\ldots,x_L;s_1,\ldots,s_t) = p(s_1)f_{s_1}(d_1)P(x_1,\ldots,x_{d_1} \mid s_1)\prod_{i=2}^{t} a_{s_{i-1}s_i} f_{s_i}(d_i)P(x_{q+1},\ldots,x_{q+d_i} \mid s_i)$$

We have $P(\phi \mid X) = \frac{P(\phi,X)}{P(X)} = \frac{P(\phi,X)}{\sum_{\phi} P(\phi,X)}$

for all ϕ on a sequence X of length L.

Viterbi decoding for GHMM



The task: compute $\underset{(s_1,\ldots,s_L)}{\operatorname{argmax}}\ p(s_1,\ldots,s_L;x_1,\ldots,x_L)$

Let the states be $\{1,\ldots,m\}$

Idea: for $i=1,\ldots,L$ and for each state l, compute:

$v_l(i)$ = the probability $p(s_1,..,s_i;x_1,..,x_i|s_i=l)$ of a most probable path up to i, which ends in state l.

Viterbi decoding for GHMM

$v_l(i)$ = the probability $p(s_1,..,s_i;x_1,..,x_i|s_i=l)$ of a most probable path up to i, which ends in state l.

For $i = 1,\ldots,L$ and for each state l we have:

$$V_l(i) = \max \begin{cases} \max\limits_{\substack{1\le q<i \\ 1\le k\le m, k\ne l}} P(x_{q+1}x_{q+2}\ldots x_i \mid s_l)V_k(q)a_{kl} \\ P(x_1x_2\ldots x_i \mid s_l)a_{0l} \end{cases}$$

Complexity: $O(m^2L^2)$

Example: a fair casino problem

HMM: hidden states {F(air), L(oaded)}, observation symbols {H(ead), T(ail)}

Transition probabilities

	F	L
F	0.0	1.0
L	1.0	0.0

Emission probabilities

	H	T
F	1/2	1/2
L	0.9	0.1

Initial prob.

P(F)=P(L)=1/2

Length distribution

	2	3
F	1/2	1/2
L	0.9	0.1

Probability of other length: 0

Find the most likely hidden state sequence for the observation sequence: HHHH

S*=FFLL or LLFF