Dishonest casino as an HMM

- \( N = 2, S = \{F, L\} \)
- \( M = 2, O = \{h, t\} \)
- \( A = \)

\[
\begin{bmatrix}
0.90 & 0.10 \\
0.05 & 0.95 \\
\end{bmatrix}
\]

- \( B = \)

\[
\begin{bmatrix}
0.9 & 0.1 \\
0.5 & 0.5 \\
0.1 & 0.9 \\
\end{bmatrix}
\]

- \( \pi = [1 \ 0] \)
A generative model for CpG islands

- There are two hidden states: CpG and non-CpG. Each state is characterized by emission probabilities of the 4 bases. You can't see which state the model is, only the emitted bases are visible.

![Diagram of hidden states and observables]

(c) Devika Subramanian, 2009
Filtering or the forward computation

- Given an HMM model \((A,B,\pi)\), and an observation sequence \(o_1, \ldots, o_t\), can we find the most likely hidden state at time \(t\), \(S_t\)?

\[ P(S_t|o_1, \ldots, o_t): \text{filtering} \]

Observation sequence: \(h \ h \ t \ t \ t \ t \ t \)

What is the hidden state here (F or L)?
What is the distribution of $S_1$? Since, $s_0=F$, we can say that $P(S_1|S_0)=[0.95 \ 0.05]$, based on the transition probabilities alone. But is that all we know?
More filtering

We have also observed $h$ at time 1. How can we fold it in into the assessment of the distribution of $S_1$?
Filtering (contd.)

\[ P(S_1 \mid o_1) = \frac{P(o_1 \mid S_1)P(S_1)}{P(o_1)} \]

\[ P(S_1 = F \mid o_1 = h) = \alpha P(h \mid F)0.95 = \alpha(0.5)(0.95) \]

\[ P(S_1 = L \mid o_1 = h) = \alpha P(h \mid L)0.05 = \alpha(0.1)(0.05) \]

\[ \alpha(0.5)(0.95) + \alpha(0.1)(0.05) = 1 \]

Therefore, \( P(S_1) = [0.99 \ 0.01] \)
Filtering computation

\[ P(S_t \mid o_t, o_1 \ldots o_{t-1}) = P(o_t \mid S_t) \sum_{s_{t-1}} P(S_t \mid s_{t-1}) P(s_{t-1} \mid o_1 \ldots o_{t-1}) \]

Recursively computed
Summary: filtering

Find \( P(S_t \mid o_1,\ldots,o_t) = cP(S_t, o_1,\ldots,o_t) \).

Define \( \alpha_t(i) = P(o_1,\ldots,o_t, S_t = s_i) \).

Initialize: \( \alpha_0(i) = \pi_i \), \( 1 \leq i \leq n \)

Recursion: \( \alpha_{t+1}(j) = b_j(o_{t+1}) \sum_{i=1}^{n} \alpha_t(i)a_{ij}, \ 0 \leq j \leq n, 1 \leq t \leq T-1 \)

Termination: \( \alpha_T(i), \ 1 \leq i \leq n \)

Time complexity \( O(n^2T) \)
Smoothing/posterior decoding

Question: can we re-estimate the distribution at $S_k$ where $k < t$, using information about the observed sequence upto time $t$?

That is, what is $P(S_k|o_1...o_t)$?
Backward computation

\[ P(S_k \mid o_1, \ldots, o_t) = cP(o_{k+1}, \ldots, o_t \mid S_k)P(S_k \mid o_1, \ldots, o_k) \]

Define \( \beta_k(i) = P(o_{k+1}, \ldots, o_t \mid S_k = s_i) \).

Initialize: \( \beta_T(i) = 1, \ 1 \leq i \leq N \).

Recursion: \( \beta_k(i) = c \sum_{j=1}^{N} a_{ij} b_j(o_{k+1}) \beta_{k+1}(j), 1 \leq i \leq N, T - 1 \leq k \leq 1 \)

Time complexity: \( O(n^2T) \)
Posterior decoding

\[ P(S_k = i \mid o_1, \ldots, o_t) = c \beta_k(i) \alpha_k(i) \]
Full Decoding

- Given HMM model \((A, B, \pi)\), and an observation sequence \(o_1 \ldots o_t\), can we find the most likely hidden state sequence \(s_1 \ldots s_t\)?
- \(\text{argmax}_{\{s_1 \ldots s_t\}} P(s_1 \ldots s_t | o_1 \ldots o_t)\)
The Viterbi algorithm

\[ \delta_t(i) = \max_{s_1,...,s_{t-1}} P(s_1,...,s_{t-1}, S_t = i, o_1,...,o_t) \]

Initialize: \( \delta_0(i) = \pi_i, 1 \leq i \leq n \)

Recursion: \( \delta_{t+1}(j) = \max_i \delta_t(i) a_{ij} b_j(o_{t+1}), \)

\[ 1 \leq t \leq T - 1, 1 \leq j \leq n \]

Computational complexity = \( O(Tn^2) \)
Learning an HMM: case 1

- Given observation sequences, and the corresponding hidden state sequences, can we find the most likely model \((A,B,\pi)\) which generated it?

Training data
Parameter estimation

- Initial state distribution
  - Fraction of times state i is state 1 in training data

- Transition probabilities
  - $a_{ij} = \frac{\text{number of transitions from } i \text{ to } j}{\text{number of transitions from } i}$

- Emission probabilities
  - $b_k(i) = \frac{\text{number of times } k \text{ is emitted in state } i}{\text{number of times state } i \text{ occurs}}$
Learning an HMM: case 2

- Given just the observation sequences, can we find the most likely model $\lambda = (A, B, \pi)$ which generated it?

$$\arg\max_{\lambda} P(o_1...o_t \mid \lambda)$$

Annotated training data is difficult to get; so we would like to derive model parameters from observable sequences.
The EM algorithm

1. Guess a model $\lambda$
2. Use observation sequence to estimate transition probabilities, emission probabilities, and initial state probabilities.
3. Update model
4. Repeat 2 and 3 till no change in model
Re-estimating parameters

- What is the probability of being in state $i$ at time $t$ and moving to state $j$, given the current model and the observation sequence $O$?

$$
\xi_t(i, j) = P(S_t = i, S_{t+1} = j \mid O, \lambda)
$$
Using forward and backward computation

\[ \xi_t(i, j) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)} \]
Re-estimating $a_{ij}$

The transition probabilities $a_{ij}$ can be re-estimated as follows

$$
\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{i=1}^{T-1} \sum_{j'=1}^{n} \xi_t(i, j')}
$$
Initial state probabilities

\[ \gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \]

Initial state probabilities are simply \( \gamma_1(i) \)

Expected number of times in state i.
Emission probabilities

\[ \hat{b}_i(k) = \frac{\text{expected number of times in state } i \text{ and observe symbol } k}{\text{expected number of times in state } i} \]

\[ \hat{b}_i(k) = \frac{\sum_{t=1}^{T} \gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \]
The EM algorithm

1. Guess a model \( \lambda = (a, b, \pi) \)
2. Use observation sequence to estimate \( \xi_t(i, j) \) and \( \gamma_t(i) \)
3. Use these estimates to recalculate \( \lambda' = (a', b', \pi') \)
4. Repeat 2 and 3 till no change in model
Summary of CpG island HMM

- Given a DNA region \( x \), Viterbi decoding predicts locations of CpG islands on it.
- Given a nucleotide \( x_i \), Viterbi decoding tells whether \( x_i \) is in a CpG island in the most likely sequence.
- Posterior decoding can assign locally optimal predictions of CpG islands.
- A fully annotated training data set can be used to estimate the generating HMM.
- Even without annotations, we can use the EM procedure to derive model parameters.
How to design an HMM for a new problem

- **Architecture/topology design:**
  - What are the states, observation symbols, and the topology of the state transition graph?

- **Learning/Training:**
  - Fully annotated or partially annotated training datasets
  - Parameter estimation by maximum likelihood or by EM

- **Validation/Testing:**
  - Fully annotated testing datasets
  - Performance evaluation (accuracy, specificity and sensitivity)
What is the probability of staying with the fair coin for T time steps?
Inherent limitation of HMMs

- The duration in state F follows an exponentially decaying distribution called a geometric distribution.

\[ P(X = F^T) = (0.95)^{T-1}(0.05) \]

- The geometric distribution gives too much probability to short sequences of Fs and Ls and too little to medium and long sequences of Fs and Ls.
Duration modeling

- To obtain non-geometric length distributions, we use an array of $n$ F states, as follows:

\[
P(\mid X \mid = L) = \binom{L-1}{n-1} p^{L-n} (1 - p)^n
\]

- Generated length distribution is a negative binomial.
Why does this matter?

- Length of stay in “Exon” state determines length of predicted exons. Very short exons are rare.

- Similarly for introns. Introns shorter than 30 bp do not exist.
Length distributions of exons and introns

Length distributions of human introns and initial, internal and terminal exons

(a) Introns

(b) Initial exons

(c) Internal exons

(d) Terminal exons
Generalized HMMs (semi-Markov HMMs)

- Each state has a specified length distribution.

Pick a state to start at $t=1$.

Repeat

- Pick the length of stay ($d$) in current state from distribution $P$.
- Emit $d$ symbols in current state.
- Pick a new state (according to a matrix) and transition to it at time $t+d$
Example

Multiple symbols emitted in each state. One to one mapping between symbols and hidden states is lost in the generalized HMM.
Viterbi algorithm for gHMMs

- Just like Viterbi for HMMs, but we use the entire stay in state instead of a state at a given time.

\[ \delta_t(i) = \max_{k=0..t-1} \max_{j \neq i} f_{t,i}(k, j) \]

Probability of most likely path ending at \( t \) with stay of \( k+1 \) in state \( i \) following a stay in state \( j \)

\[ f_{t,i}(k, j) = \prod_{r=0}^{k} b_i(o_{t-r}) l_i(k) a_{ji} \delta_{t-k-1}(j) \]