Hidden Markov Models

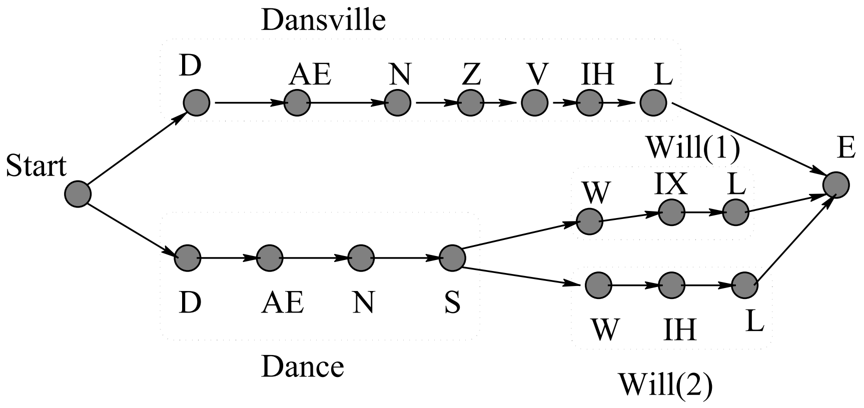
Adapted from

Dr Catherine Sweeney-Reed’s slides

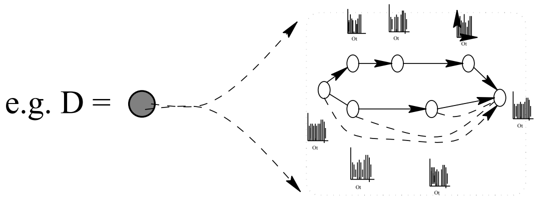
Speech Recognition

1. Build HMMs for each word in the corpus

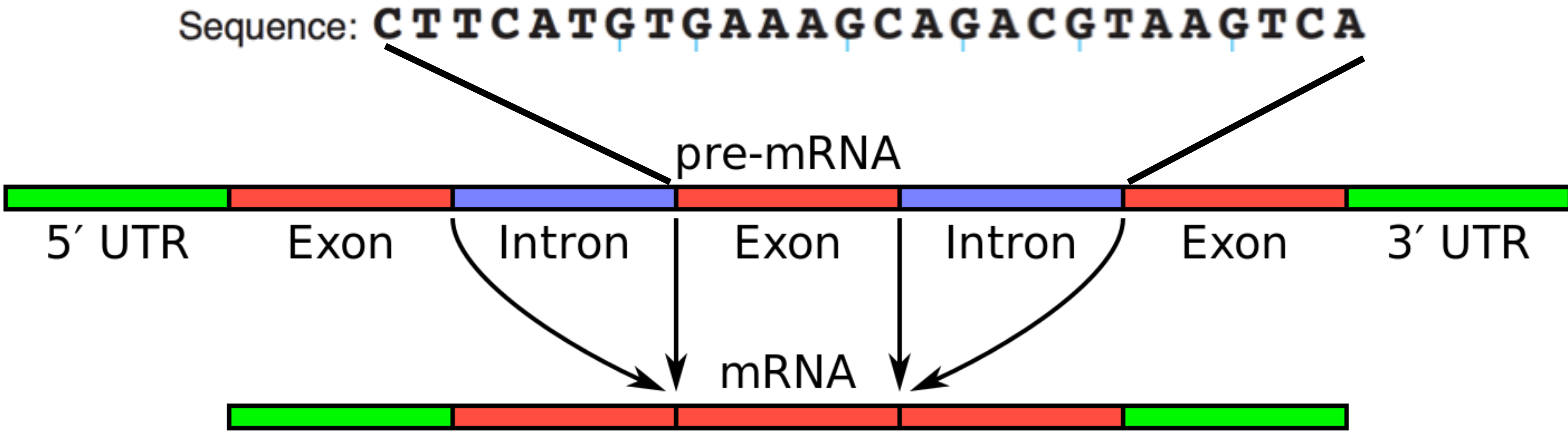
2. For a given word, W find the HMM that grades W most probable



Also works for phonemes



Detailed Model For a Phoneme



Let’s imagine some simple differences: say that exons have a uniform base composition on average (25% each base), introns are A/T rich (say, 40% each for A/T, 10% each for C/G), and the 5′SS consensus nucleotide is almost always a G (say, 95% G and 5% A).

HMM rates the location of the cut point

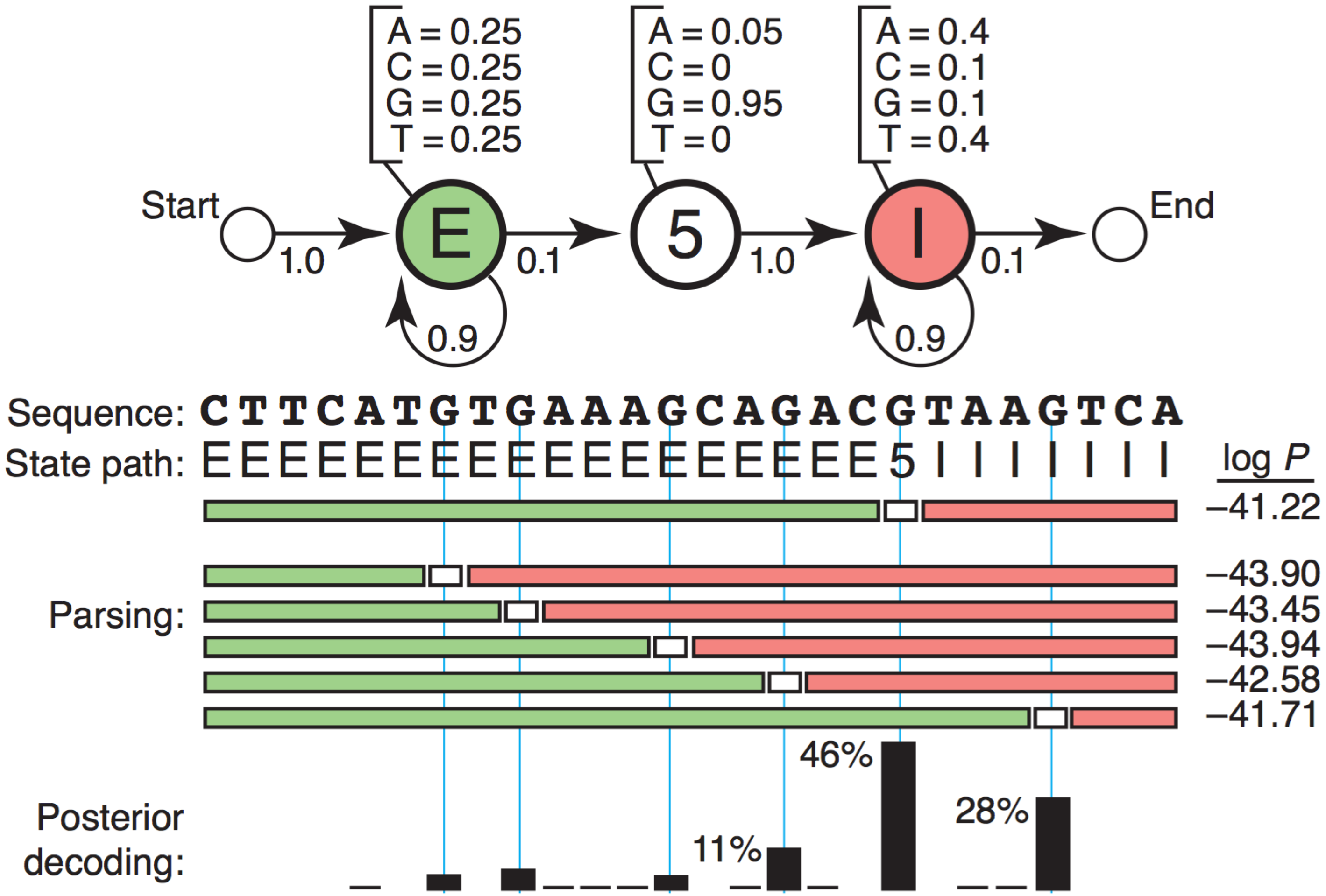
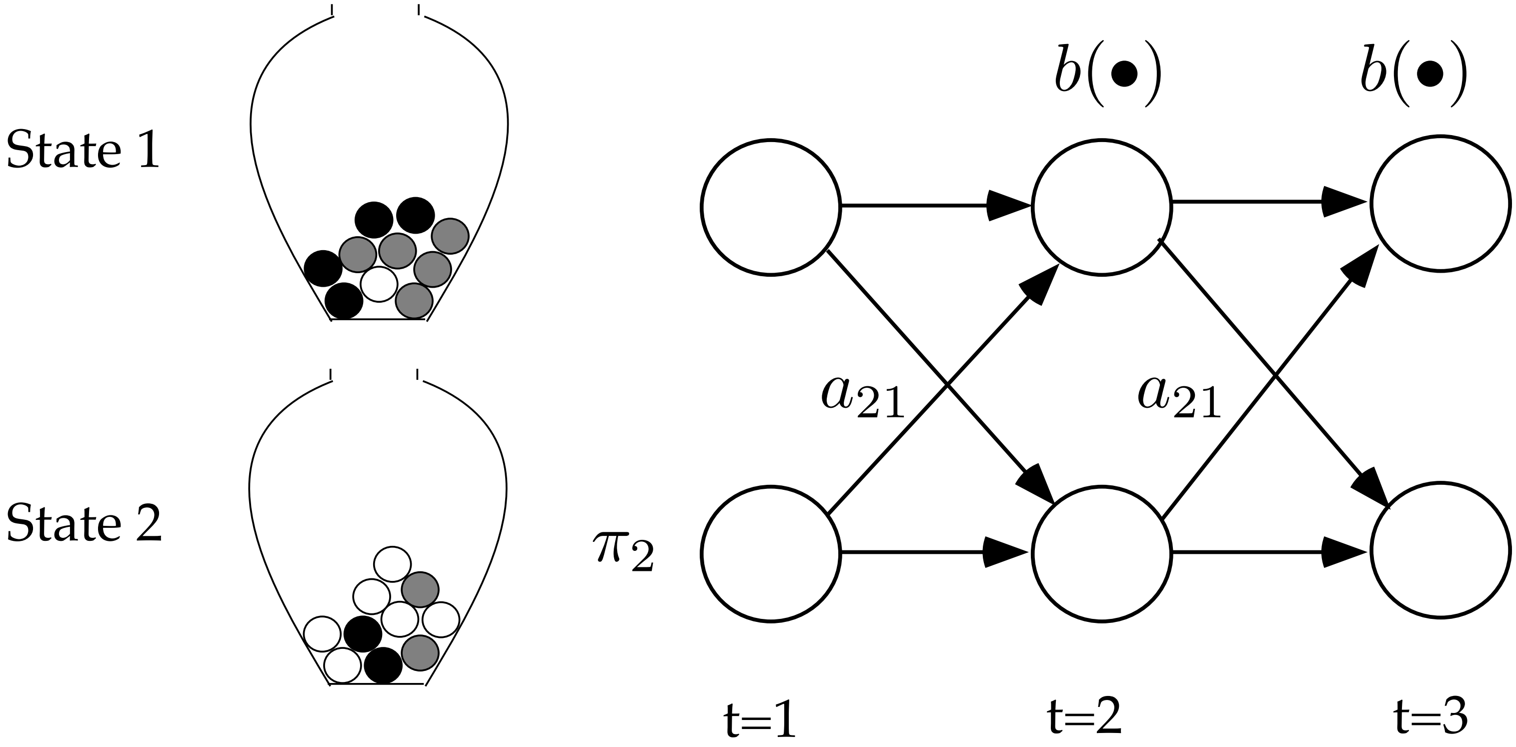


Figure 1 A toy HMM for 5′ splice site recognition. See text for explanation.

Specification of an HMM

- *N - number of states*
  - $Q = \{q_1; q_2; : : : ;q_T\}$ *- set of states*
- *M - the number of symbols (observables)*
  - $O = \{o_1; o_2; : : : ;o_T\}$ *- set of symbols*

The Urn model of a HMM



N=2
M=3
O= {black, white, gray}

Specification of an HMM

- A - *the state transition probability matrix*
  - □ $aij = P(q_{t+1} = j|q_t = i)$
- B- *observation probability distribution*
  - □ $b_j(k) = P(o_t = k|q_t = j) \quad 1 \leq k \leq M$
- π - *the initial state distribution*

Specification of an HMM

- Full HMM is thus specified as a triplet:
    - $\lambda = (A, B, \pi)$

Central problems in HMM modeling

- Problem 1

  Evaluation:
  - Probability of occurrence of a particular observation sequence, O = {o_1,…,o_k}, given the model
  - $P(O|\lambda)$
  - Complicated – hidden states
  - Useful in sequence classification

Central problems in HMM modeling

- Problem 2

  Decoding:
  - Optimal state sequence to produce given observations, $O = \{o_1,\ldots,o_k\}$, given model
  - Optimality criterion
  - Useful in recognition problems

Central problems in HMM modeling

- Problem 3

  Learning:
  - Determine optimum model, given a training set of observations
  - Find λ, such that P(O|λ) is maximal

Problem 1: Naïve solution

- State sequence Q = (q_1,…q_T)
- Assume independent observations:

$$P(O \mid q, \lambda) = \prod_{i=1}^{T} P(o_t \mid q_t, \lambda) = b_{q1}(o_1) b_{q2}(o_2) \ldots b_{qT}(o_T)$$

NB Observations are mutually independent, given the hidden states. (Joint distribution of independent variables factorizes into marginal distributions of the independent variables.)

Problem 1: Naïve solution

- Observe that :

$$P(q \mid \lambda) = \pi_{q1} a_{q1q2} a_{q2q3} \ldots a_{qT-1qT}$$

- And that:

$$P(O \mid \lambda) = \sum_{q} P(O \mid q, \lambda) P(q \mid \lambda)$$

Problem 1: Naïve solution

- So there is a solution:

$$P(O \mid \lambda) = \sum_q P(O \mid q, \lambda) P(q \mid \lambda)$$

But it is very expensive:

-The above sum is over all state paths

-There are N^T states paths, each ‘costing’
O(T) calculations, leading to $O(TN^T)$ time complexity.

Problem 1: Efficient solution

Forward algorithm:

- Define auxiliary forward variable α:

$$\alpha_t(i) = P(o_1, \ldots, o_t, q_t = i \mid \lambda)$$

$\alpha_t(i)$ is the probability of observing a partial sequence of observables $o_1, \ldots o_t$ such that at time t, state $q_t=i$

Problem 1: Efficient solution

- Recursive algorithm:
  - □ Initialise:

(Partial obs seq to *t* AND state *i* at *t*) x (transition to *j* at *t+1*) x (sensor)

  - □ Calculate: $\alpha_1(i) = \pi_i b_i(o_1)$

  - □ Obtain: $\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) a_{ij}] b_j(o_{t+1})$

Sum, as can reach *j* from any preceding state

α incorporates partial obs seq to *t*

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Sum of different ways of getting obs seq

Complexity is O(N^2T)

Problem 1: Alternative solution

Backward algorithm:

- Define auxiliary forward variable *β*:

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, \ldots, o_T, q_t = i \mid \lambda)$$

$\beta_t(i)$ the probability of observing a sequence of observables $o_{t+1}, \ldots, o_T$ given state $q_t = i$ at time *t*, and λ

Problem 1: Alternative solution

- Recursive algorithm:
  - □ Initialise: $\beta_T(j) = 1$
  - □ Calculate: $$\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} b_j(o_{t+1})$$
  - □ Terminate: $p(O \mid \lambda) = \sum_{i=1}^{N} \beta_1(i)$ $\qquad t = T-1,\ldots,1$

Complexity is $O(N^2T)$

Problem 2: Decoding

- Choose state sequence to maximize probability of observation sequence
- Viterbi algorithm - inductive algorithm that keeps the best state sequence at each instance

Problem 2: Decoding

Viterbi algorithm:

- State sequence to maximise P(O,Q|λ):

$$P(q_1, q_2, \ldots q_T \mid O, \lambda)$$

- Define auxiliary variable δ:

$$\delta_t(i) = \max_q P(q_1, q_2, \ldots, q_t = i, o_1, o_2, \ldots o_t \mid \lambda)$$

$\delta_t(i)$ – the probability of the most probable path ending in state q_t=i

Central problems

Problem 2: Decoding

To get state seq, need to keep track of argument to maximise this, for each *t* and *j*. Done via the array *$\psi_t(j)$*.

- Recurrent property:

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

- Algorithm:
  - □ 1. Initialise:

$$\delta_1(i) = \pi_i b_i(o_1) \qquad 1 \le i \le N$$

$$\psi_1(i) = 0$$

Problem 2: Decoding

Central problems

- 2. Recursion:

$$\delta_t(j) = \max_{1 \le i \le N}(\delta_{t-1}(i)a_{ij})b_j(o_t)$$

$$\psi_t(j) = \arg\max_{1 \le i \le N}(\delta_{t-1}(i)a_{ij}) \qquad 2 \le t \le T, 1 \le j \le N$$

- 3. Terminate:

P* gives the state-optimised probability

$$P^* = \max_{1 \le i \le N} \delta_T(i)$$

$$q_T^* = \arg\max_{1 \le i \le N} \delta_T(i)$$

Q* is the optimal state sequence (Q* = {q1*,q2*,…,qT*})

Problem 2: Decoding

- 4. Backtrack state sequence:

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \qquad t + T - 1, T - 2, \ldots, 1$$

O(N^2T) time complexity

Expectation Maximization

A classification algorithm has parameters

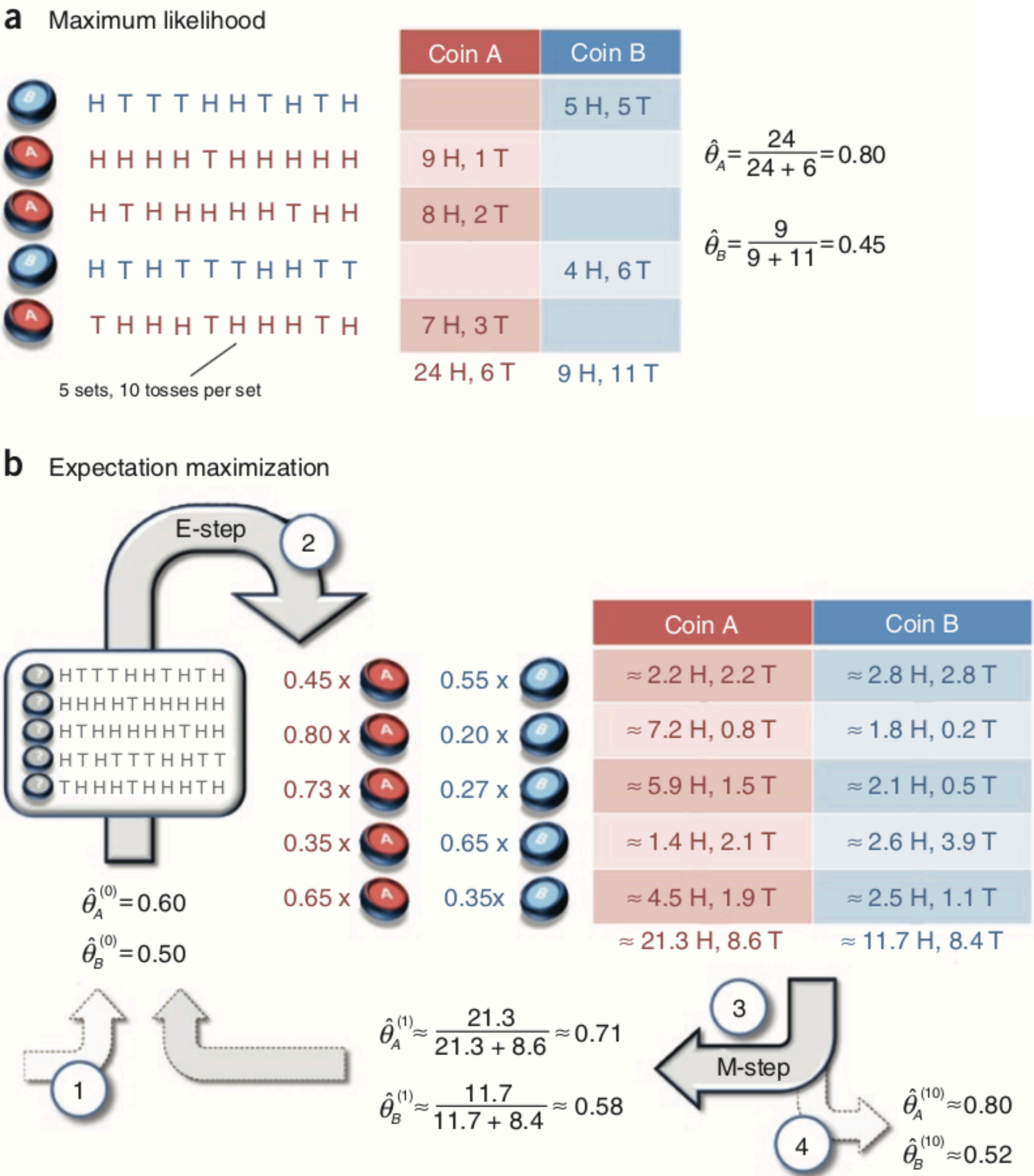
1. Use it do classify data

2. Use the classified data to re-estimate parameters

Repeat steps 1 & 2 until parameter estimates converge

Ex 1
data from two
coin flips has their
coin IDs lost.

Use the data to re-estimate the
coin probabilities



		Coin A	Coin B
B	H T T T H H T H T H		5 H, 5 T
A	H H H H T H H H H H	9 H, 1 T	
A	H T H H H H H T H H	8 H, 2 T	
B	H T H T T T H H T T		4 H, 6 T
A	T H H H T H H H T H	7 H, 3 T	
		24 H, 6 T	9 H, 11 T

$\hat{\theta}_A = \frac{24}{24+6} = 0.80$

$\hat{\theta}_B = \frac{9}{9+11} = 0.45$

$\hat{\theta}_A^{(0)} = 0.60$

$\hat{\theta}_B^{(0)} = 0.50$

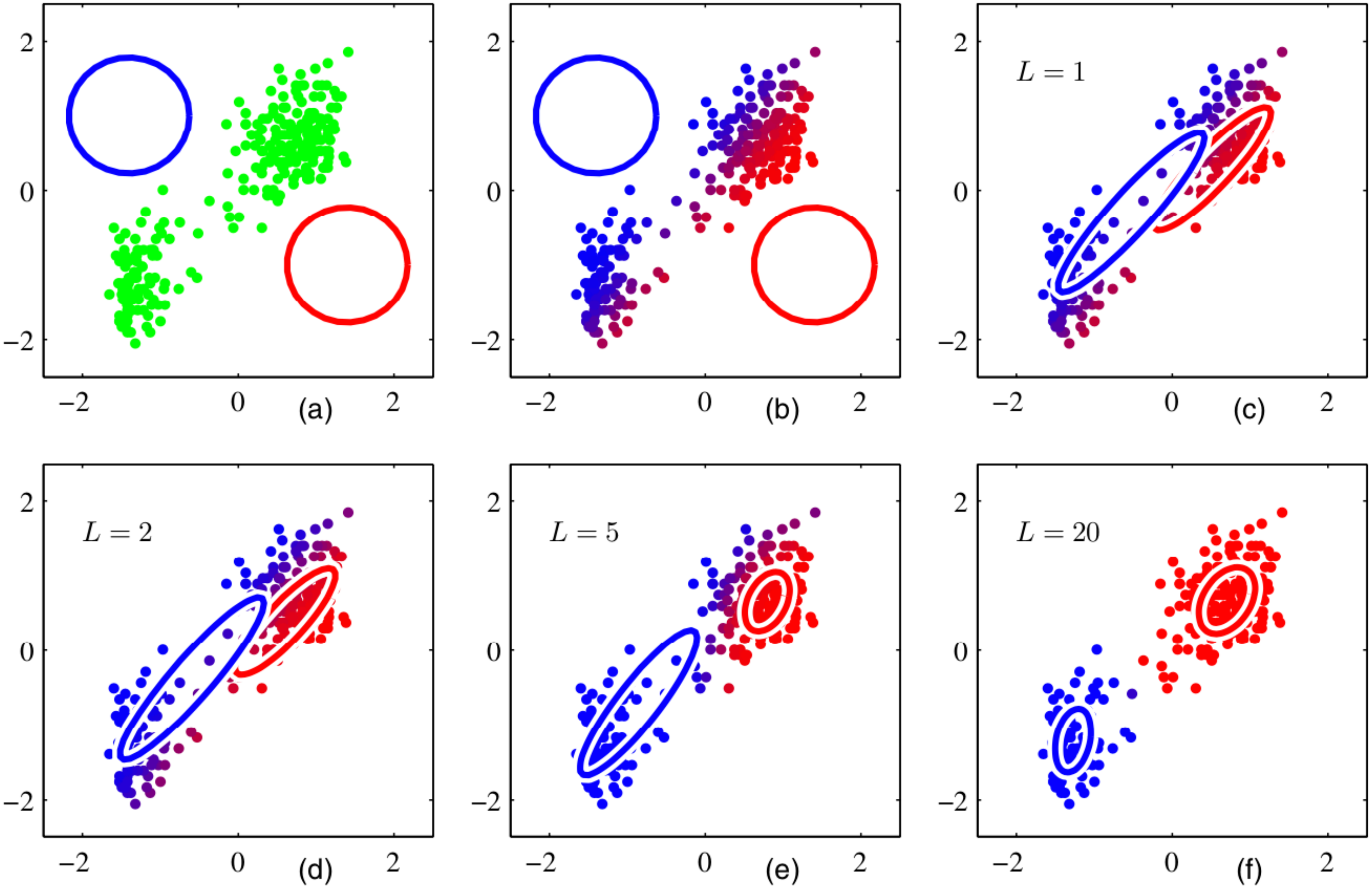
			Coin A	Coin B
H T T T H H T H T H	0.45 x A	0.55 x B	≈ 2.2 H, 2.2 T	≈ 2.8 H, 2.8 T
H H H H T H H H H H	0.80 x A	0.20 x B	≈ 7.2 H, 0.8 T	≈ 1.8 H, 0.2 T
H T H H H H H T H H	0.73 x A	0.27 x B	≈ 5.9 H, 1.5 T	≈ 2.1 H, 0.5 T
H T H T T T H H T T	0.35 x A	0.65 x B	≈ 1.4 H, 2.1 T	≈ 2.6 H, 3.9 T
T H H H T H H H T H	0.65 x A	0.35x B	≈ 4.5 H, 1.9 T	≈ 2.5 H, 1.1 T
			≈ 21.3 H, 8.6 T	≈ 11.7 H, 8.4 T

$\hat{\theta}_A^{(1)} \approx \frac{21.3}{21.3+8.6} \approx 0.71$

$\hat{\theta}_B^{(1)} \approx \frac{11.7}{11.7+8.4} \approx 0.58$

$\hat{\theta}_A^{(10)} \approx 0.80$

$\hat{\theta}_B^{(10)} \approx 0.52$

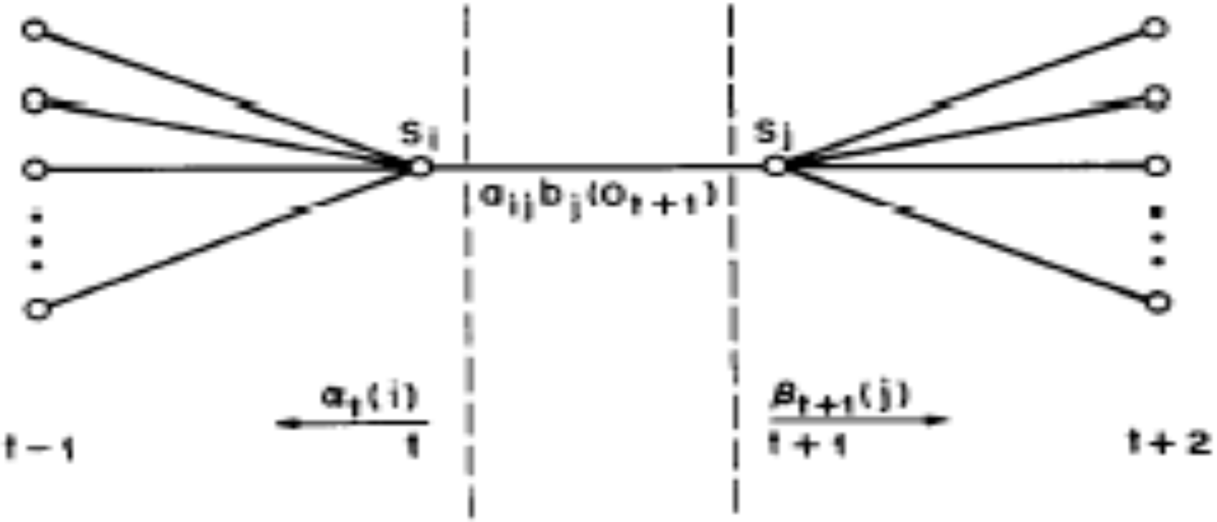

2
0
−2
−2
0
2
(a)
(b)
L = 1
(c)
L = 2
(d)
L = 5
(e)
L = 20
(f)

Problem 3: Learning

- Training HMM to encode obs seq such that HMM should identify a similar obs seq in future
- Find λ=(A,B,π), maximizing P(O|λ)
- General algorithm:
  - □ Initialise: λ_0
  - □ Compute new model λ, using λ_0 and observed sequence O
  - □ Then $\lambda_o \leftarrow \lambda$
  - □ Repeat steps 2 and 3 until:

$$\log P(O \mid \lambda) - \log P(O \mid \lambda_0) < d$$

Problem 3: Learning



Operations required for the computation of the joint event that the system is in state Si and time t and State Sj at time t+1

Central problems

Problem 3: Learning

- Let $\gamma_t(i)$be a probability of being in state *i* at time *t*, given *O*

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

- $\sum_{t=1}^{T-1} \gamma_t(i)$ - expected no. of transitions from state *i*
- $\sum_{t=1}^{T-1} \xi_t(i)$ - expected no. of transitions $i \rightarrow j$

Problem 3: Learning

Step 2 of Baum-Welch algorithm:

- $\hat{\pi} = \gamma_1(i)$ the expected frequency of state *i* at time *t=1*
- $\hat{a}_{ij} = \frac{\sum \xi_t(i,j)}{\sum \gamma_t(i)}$ ratio of expected no. of transitions from state *i* to *j* over expected no. of transitions from state *i*
- $\hat{b}_j(k) = \frac{\sum_{t,o_t=k} \gamma_t(j)}{\sum \gamma_t(j)}$ ratio of expected no. of times in state *j* observing symbol *k* over expected no. of times in state *j*

Central problems

Problem 3: Learning

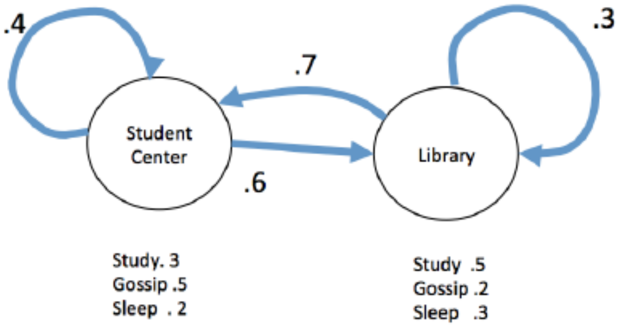
- Baum-Welch algorithm uses the forward and backward algorithms to calculate the auxiliary variables α,β
- B-W algorithm is a special case of the EM algorithm:
  - □ E-step: calculation of ξ and γ
  - □ M-step: iterative calculation of $\hat{\pi}$, $\hat{a}_{ij}$, $\hat{b}_j(k)$
- Practical issues:
  - □ Can get stuck in local maxima
  - □ Numerical problems – log and scaling

2. **HMMs**

You are in charge of mentoring your roommate, who has big exam coming up. She reports that she may have gossiped on the last day but she has been studying on the other two and has been in the library for the last three days. To help you keep track, you have built the following Markov model of her behavior that has initial probabilities:

$$p(\text{Library}, \text{Center}) = (.6, .4)$$

and structure given by:



(a) [5] Draw the three-day HMM

(b) [20]Is your roommate's report the most likely scenario? Show all steps you use to solve this problem

Further Reading

- 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.
- R. Dugad and U. B. Desai, "A tutorial on Hidden Markov models," Signal Processing and Artifical Neural Networks Laboratory, Dept of Electrical Engineering, Indian Institute of Technology, Bombay Technical Report No.: SPANN-96.1, 1996.
- W.H. Laverty, M.J. Miket, and I.W. Kelly, “Simulation of Hidden Markov Models with EXCEL”, The Statistician, vol. 51, Part 1, pp. 31-40, 2002