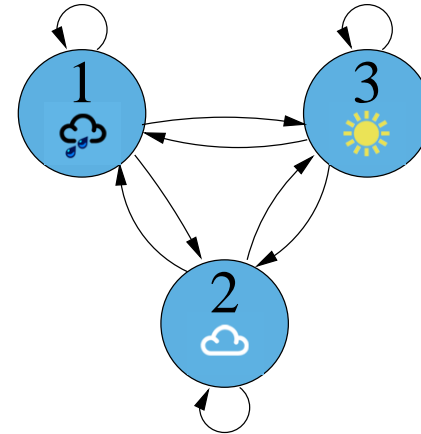


HMM part 1

Dr Philip Jackson

- Pattern matching of sequences
- Probability fundamentals
- Markov models
- Hidden Markov models
 - Likelihood calculation



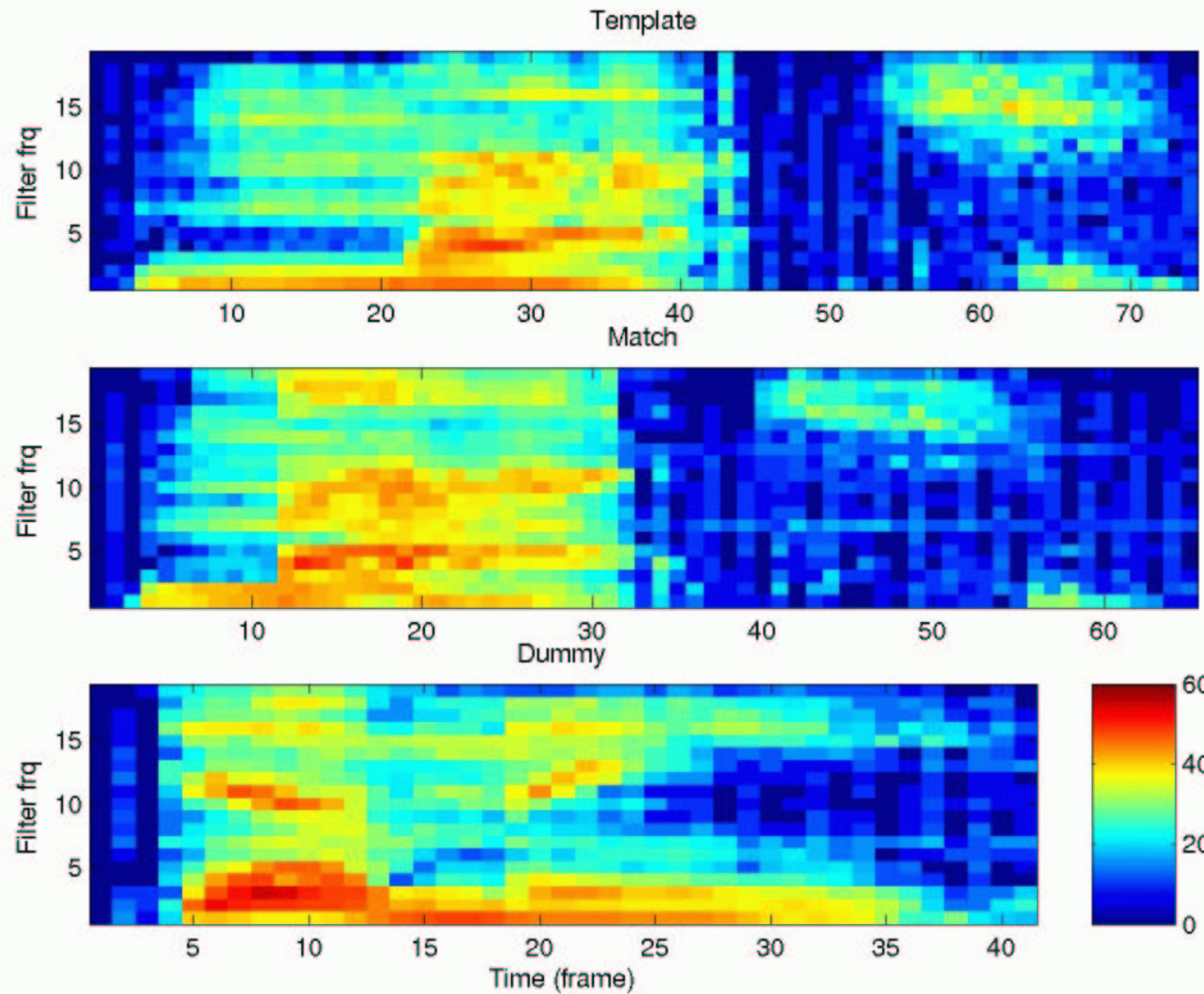
Approaches to recognizing sequences

- Rule-based heuristics
- Pattern matching
 1. **dynamic time warping** (deterministic)
 2. **Hidden Markov models** (stochastic)
- Classification
 1. **artificial neural networks** (discriminative)
- Applications: automatic speech recognition, character recognition, protein and DNA sequencing, speech synthesis, noise-robust data transmission, cryptoanalysis, machine translation, image classification, etc.

Distance-from-template pattern matching

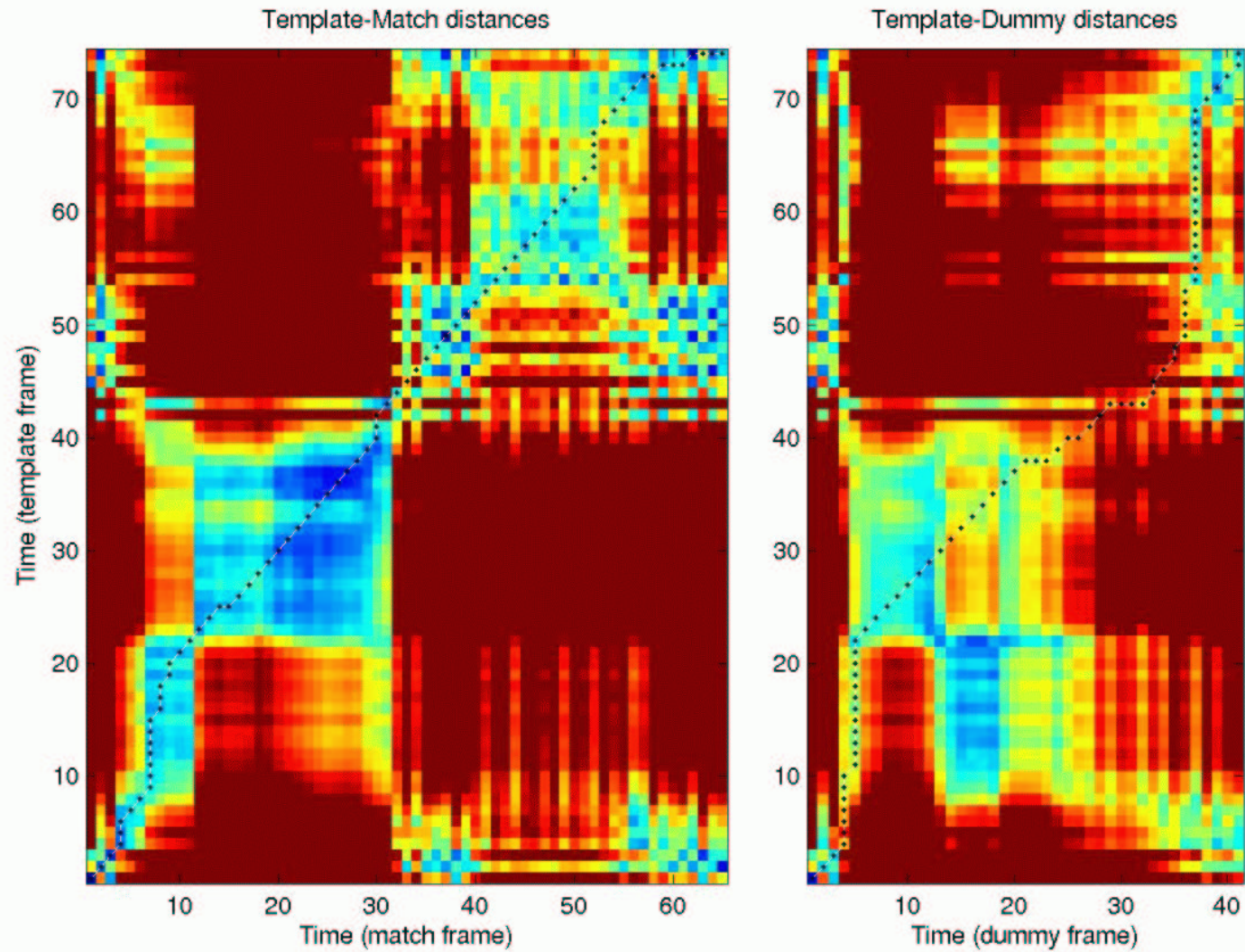
- Template
 - the features of a typical example of the sequence to be recognized
 - e.g., filterbank, linear prediction/PLP, cepstrum/MFCC
- Distance
 - a measure of how well the features of a new test sequence match those of the reference template
 - e.g., Euclidean distance, Mahalanobis distance, Itakura distance

Utterance features



Template and test word features: “match”, “match” & “dummy”.

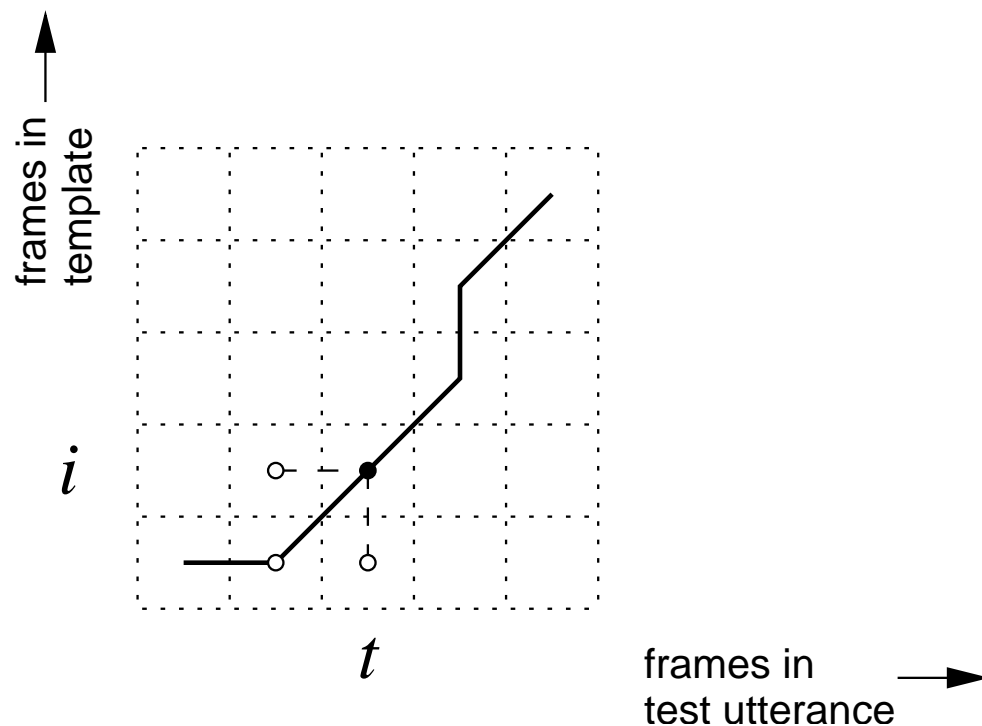
Inter-utterance distances



Computed Euclidean feature distances for template and test words.

Dynamic time warping

DTW is a method of pattern matching that allows for timescale variations in sequences of the same class.



Time alignment of two instances of the same word (Holmes & Holmes 2001, p.116). Open circles mark permitted predecessors to the closed circle at (t, i) .

Dynamic programming for time alignment

Cummulative distance along the best path upto frame i in template and t th test frame is:

$$D(t, i) = \sum_{u,v=1,1}^{t,i} \Big|_{\text{along best path}} d(u, v), \quad (1)$$

where $d(u, v)$ is distance between features from u th frame of test utterance and those from v th frame of template.

If we only allow transitions from current and previous frames, we have

$$D(t, i) = \min [D(t, i - 1), D(t - 1, i - 1), D(t - 1, i)] + d(t, i). \quad (2)$$

DTW algorithm

1. Initialise,

$$D(1, i) = \begin{cases} d(1, i) & \text{for } i = 1 \\ d(1, i) + D(1, i - 1) & \text{for } i = 2, \dots, N \end{cases} \quad (3)$$

2. Recur for $t = 2, \dots, T$,

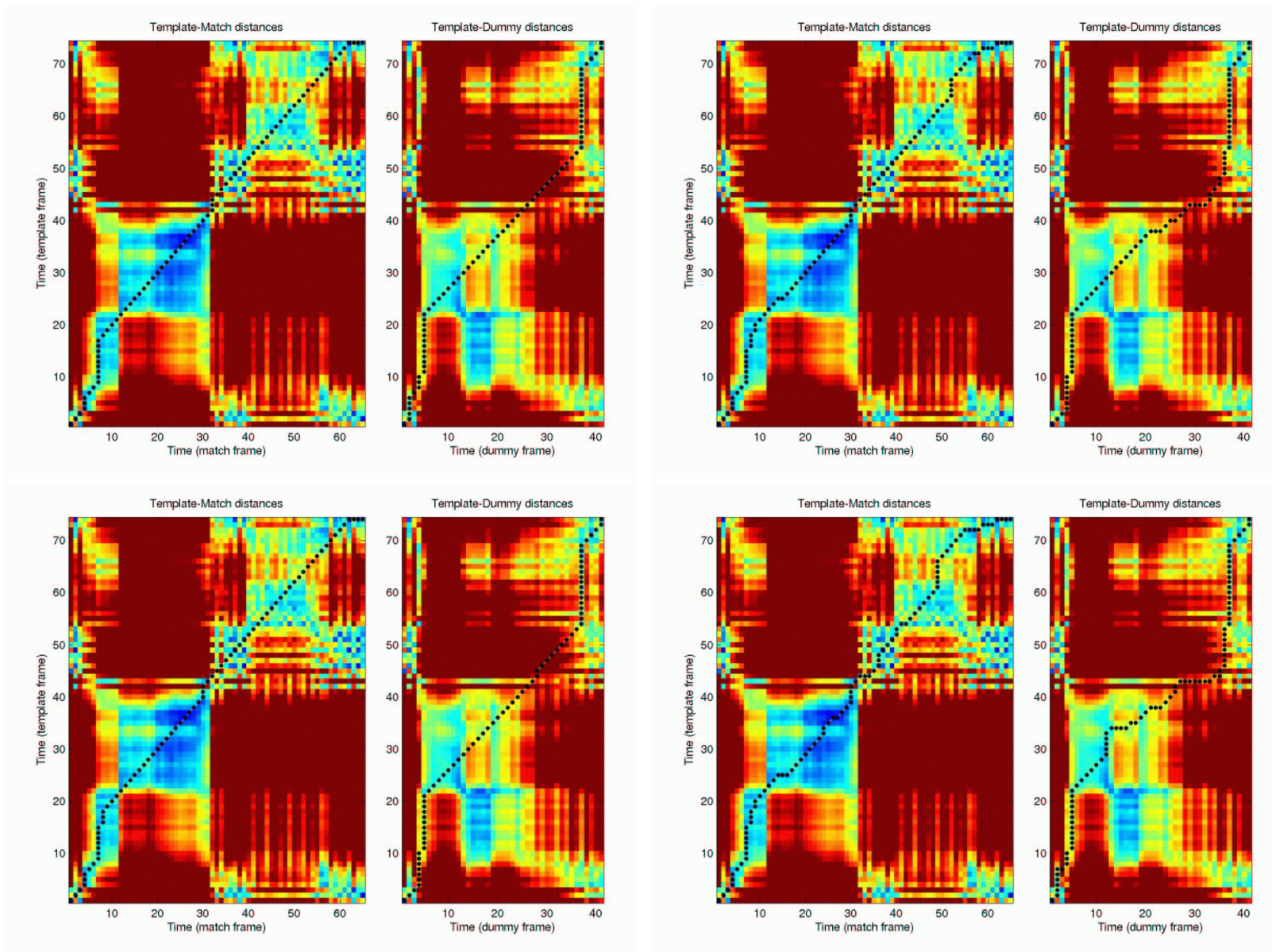
$$D(t, i) = \begin{cases} d(t, i) + D(t - 1, i) & \text{for } i = 1 \\ d(t, i) + \min [D(t, i - 1), \\ \quad D(t - 1, i - 1), \\ \quad D(t - 1, i)] & \text{for } i = 2, \dots, N \end{cases} \quad (4)$$

3. Finalise,

$$\Delta = D(T, N).$$

Thus, every possible path's cost is evaluated recursively.

Examples of DTW with distortion penalty



Path with various distortion penalties (clockwise from top left): none, standard, low and high.

Summary of Dynamic Time Warping

- Problems:

1. How many templates should we register?
2. How do we select the best ones?
3. How do we determine a fair distance metric?
4. How should we set the distortion penalties?

- Solution:

- Develop an inference framework to build templates based on the statistics of our data.

Probability fundamentals

Normalisation

Discrete: probability of all possibilities sums to one:

$$\sum_{\text{all } X} P(X) = 1. \quad (5)$$

Continuous: integral over entire probability density function (pdf) comes to one:

$$\int_{-\infty}^{\infty} p(x) dx = 1. \quad (6)$$

Joint probability

The joint probability that two independent events occur is the product of their individual probabilities:

$$P(A, B) = P(A) P(B). \quad (7)$$

Conditional probability

If two events are dependent, we need to determine their conditional probabilities. The joint probability is now

$$P(A, B) = P(A) P(B|A), \quad (8)$$

where $P(B|A)$ is the probability of event B given that A occurred; conversely, taking the events the other way

$$P(A, B) = P(A|B) P(B). \quad (9)$$

	A	\bar{A}	
B	0.1	0.3	0.4
\bar{B}	0.4	0.2	0.6
	0.5	0.5	1.0

These expressions can be rearranged to yield the conditional probabilities. Also, we can combine them to obtain the theorem proposed by Rev. Thomas Bayes (C.18th).

Bayes' theorem

Equating the RHS of eqs. 8 and 9 gives

$$P(B|A) = \frac{P(A|B) P(B)}{P(A)}. \quad (10)$$

For example, in a word recognition application we have

$$P(w|\mathcal{O}) = \frac{p(\mathcal{O}|w) P(w)}{p(\mathcal{O})}, \quad (11)$$

which can be interpreted as

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}. \quad (12)$$

The *posterior* probability is used to make Bayesian inferences; the conditional *likelihood* describes how likely the data were for a given class; the *prior* allows us to incorporate other forms of knowledge into our decision (like a language model); the *evidence* acts as a normalisation factor and is often discarded in practice (as it is the same for all classes).

Marginalisation

Discrete: probability of event B , which depends on A , is the sum over A of all joint probabilities:

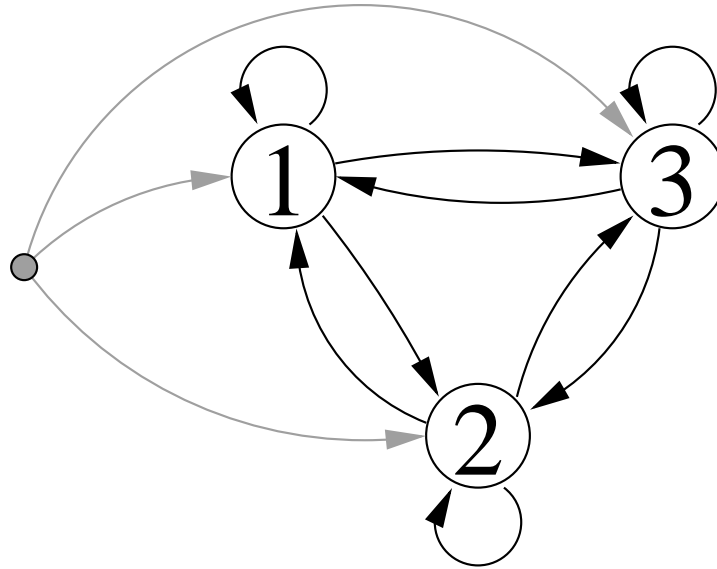
$$P(B) = \sum_{\text{all } A} P(A, B) = \sum_{\text{all } A} P(B|A) P(A). \quad (13)$$

Continuous: similarly, the nuisance factor x can be eliminated from its joint pdf with y :

$$p(y) = \int_{-\infty}^{\infty} p(x, y) dx = \int_{-\infty}^{\infty} p(y|x)p(x) dx \quad (14)$$

Introduction to Markov models

State topology of an ergodic Markov model:



The initial-state probabilities for each state i are defined

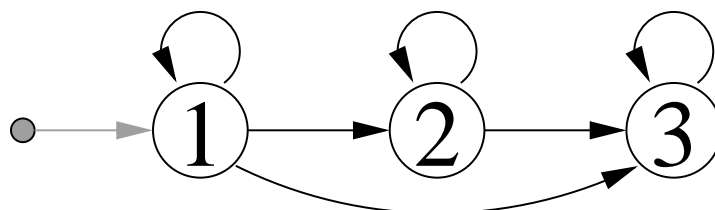
$$\pi_i = P(x_1 = i), \quad 1 \leq i \leq N. \quad (15)$$

with the properties

$$\pi_i \geq 0, \quad \text{and} \quad \sum_{i=1}^N \pi_i = 1, \quad \forall i.$$

Modeling stochastic sequences

State topology of a left-right Markov model:



For 1st-order Markov chains, probability of state occupation depends only on the previous step (Rabiner, 1989):

$$P(x_t = j | x_{t-1} = i, x_{t-2} = h, \dots) \approx P(x_t = j | x_{t-1} = i). \quad (16)$$

So, if we assume the RHS of eq. 16 is independent of time, we can write the state-transition probabilities as

$$a_{ij} = P(x_t = j | x_{t-1} = i), \quad 1 \leq i, j \leq N, \quad (17)$$

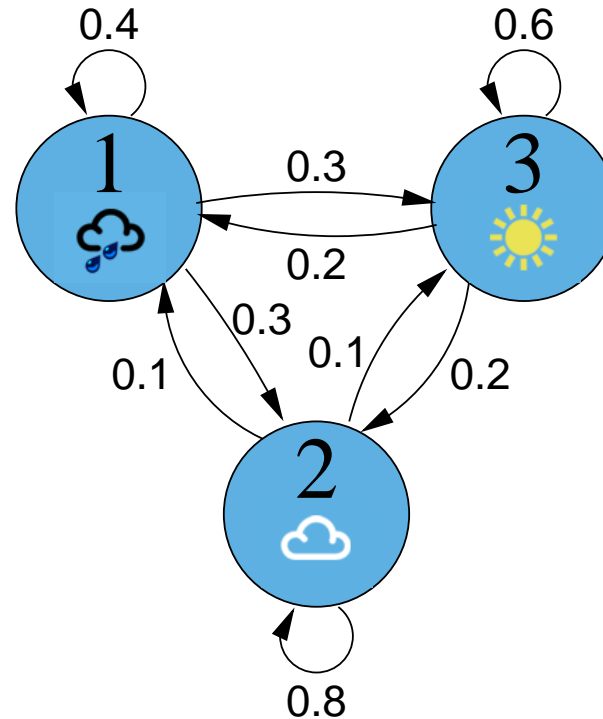
with the properties

$$a_{ij} \geq 0, \quad \text{and} \quad \sum_{j=1}^N a_{ij} = 1, \quad \forall i, j.$$

Weather predictor example

Let us represent the state of the weather by a 1st-order, ergodic Markov model, \mathcal{M} :

State 1: rain
State 2: cloud
State 3: sun



with state-transition probabilities,

$$A = \{a_{ij}\} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}. \quad (18)$$

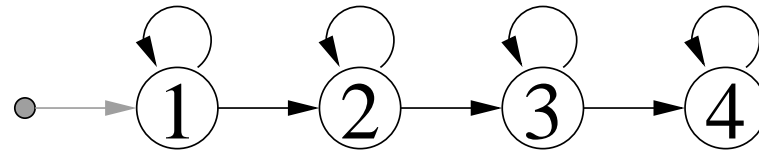
Weather predictor probability calculation

Given today is sunny (i.e., $x_1 = 3$), what is the probability with model \mathcal{M} of directly observing the sequence of weather states “sun-sun-rain-cloud-cloud-sun”?

$$\begin{aligned} P(X|\mathcal{M}) &= P(X = \{3, 3, 1, 2, 2, 3\}|\mathcal{M}) \\ &= P(x_1 = 3) P(x_2 = 3|x_1 = 3) \\ &\quad P(x_3 = 1|x_2 = 3) P(x_4 = 2|x_3 = 1) \\ &\quad P(x_5 = 2|x_4 = 2) P(x_6 = 3|x_5 = 2) \\ &= \pi_3 a_{33} a_{31} a_{12} a_{22} a_{23} \\ &= 1 \times (0.8)(0.1)(0.3)(0.6)(0.2) \\ &= 0.00288 \\ &\approx 0.3\% \end{aligned}$$

Summary of Markov models

State topology:



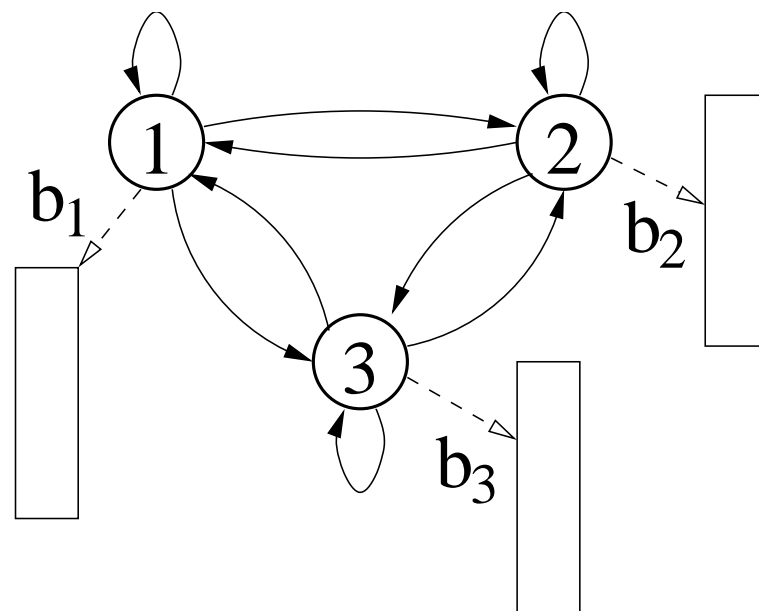
Initial-state probabilities: $\pi = \{\pi_i\} = [1 \ 0 \ 0 \ 0]$,
and state-transition probabilities:

$$A = \{a_{ij}\} = \begin{bmatrix} 0.6 & 0.4 & 0 & 0 \\ 0 & 0.9 & 0.1 & 0 \\ 0 & 0 & 0.2 & 0.8 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}.$$

Probability of a given state sequence X :

$$\begin{aligned} P(X|\mathcal{M}) &= \pi_{x_1} a_{x_1x_2} a_{x_2x_3} a_{x_3x_4} \cdots \\ &= \pi_{x_1} \prod_{t=2}^T a_{x_{t-1}x_t}. \end{aligned} \quad (19)$$

Introduction to hidden Markov models



Probability of state i generating a *discrete* observation o_t , which has one of a finite set of values, is

$$b_i(o_t) = P(o_t | x_t = i). \quad (20)$$

Probability distribution of a *continuous* observation o_t , which can have one of an infinite set of values, is

$$b_i(o_t) = p(o_t | x_t = i). \quad (21)$$

We begin by considering only discrete observations.

Elements of a discrete HMM, λ

1. Number of different states N , $x \in \{1, \dots, N\}$;

2. Number of different events K , $k \in \{1, \dots, K\}$;

3. Initial-state probabilities,

$$\pi = \{\pi_i\} = \{P(x_1 = i)\} \quad \text{for } 1 \leq i \leq N;$$

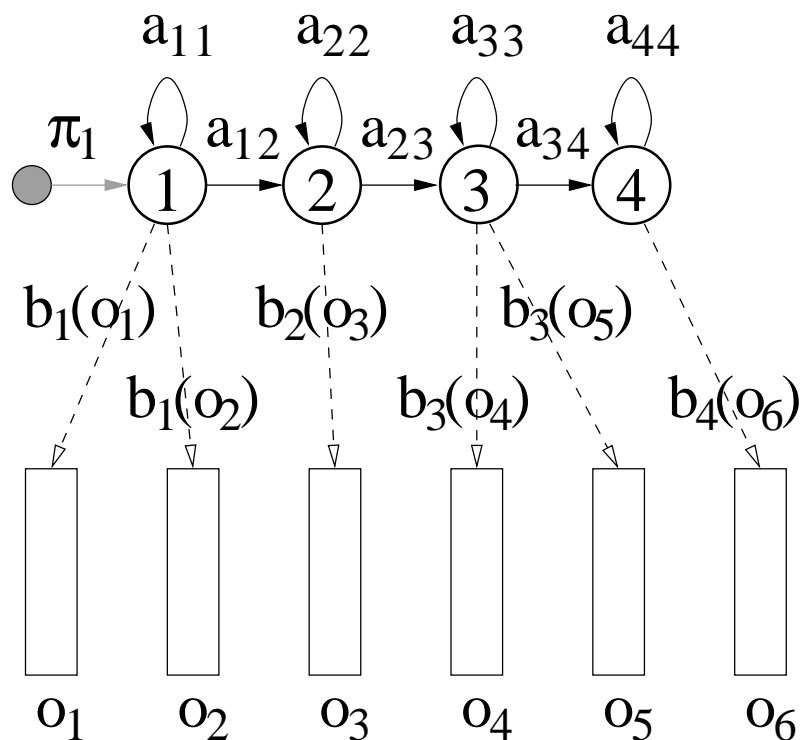
4. State-transition probabilities,

$$A = \{a_{ij}\} = \{P(x_t = j | x_{t-1} = i)\} \quad \text{for } 1 \leq i, j \leq N;$$

5. Discrete output probabilities,

$$B = \{b_i(k)\} = \{P(o_t = k | x_t = i)\} \quad \begin{array}{l} \text{for } 1 \leq i \leq N \\ \text{and } 1 \leq k \leq K. \end{array}$$

Illustration of HMM as observation generator

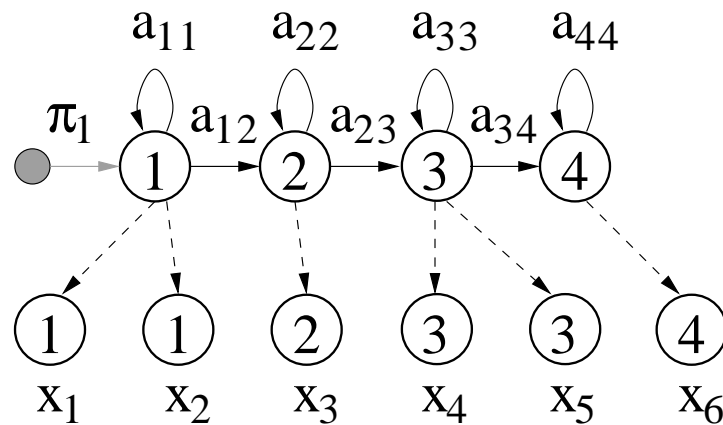


The state sequence $X = \{1, 1, 2, 3, 3, 4\}$ produces the set of observations $\mathcal{O} = \{o_1, o_2, \dots, o_6\}$:

$$\begin{aligned}
 P(X|\lambda) &= \pi_1 a_{11} a_{12} a_{23} a_{33} a_{34} \\
 P(\mathcal{O}|X, \lambda) &= b_1(o_1) b_1(o_2) b_2(o_3) b_3(o_4) b_3(o_5) b_4(o_6) \\
 P(\mathcal{O}, X|\lambda) &= P(\mathcal{O}|X, \lambda)P(X|\lambda) \\
 &= \pi_1 b_1(o_1) a_{11} b_1(o_2) a_{12} b_2(o_3) \dots \quad (22)
 \end{aligned}$$

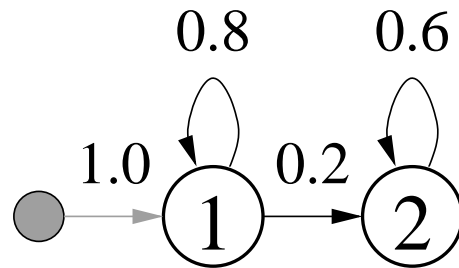
Example of the Markov Model, \mathcal{M}

- (a) Initial-state probabilities,
 $\pi = \{\pi_i\} = \{P(x_1 = i)\}$ for $1 \leq i \leq N$;
- (b) State-transition probabilities,
 $A = \{a_{ij}\} = \{P(x_t = j | x_{t-1} = i)\}$ for $1 \leq i, j \leq N$;



producing a sequence
 $X = \{1, 1, 2, 3, 3, 4\}$.

Probability of MM state sequence



Transition probabilities:

$$\pi = \{\pi_i\} = \begin{bmatrix} 1 & 0 \end{bmatrix}, \text{ and}$$

$$A = \{a_{ij}\} = \begin{bmatrix} 0.8 & 0.2 \\ 0 & 0.6 \end{bmatrix}.$$

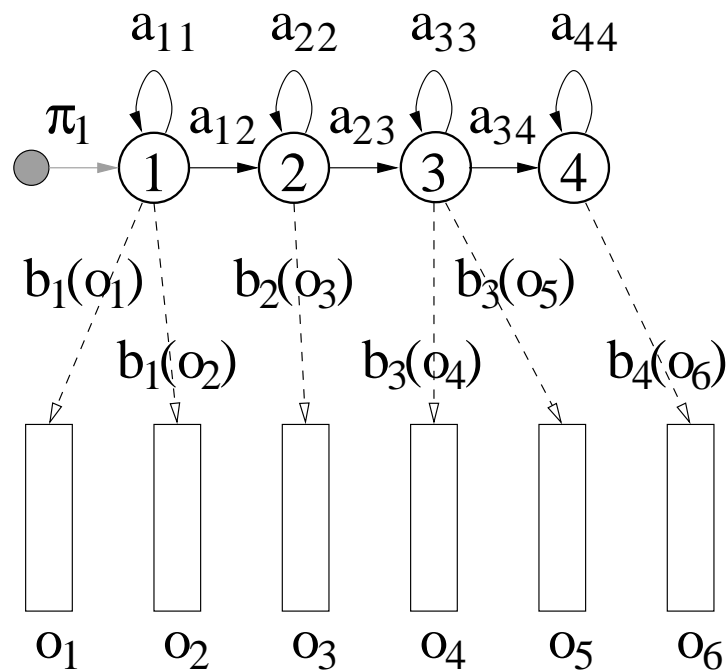
Probability of a certain state sequence, $X = \{1, 2, 2\}$:

$$\begin{aligned} P(X|\mathcal{M}) &= \pi_1 a_{12} a_{22} \\ &= 1 \times \\ &= \end{aligned}$$

(23)

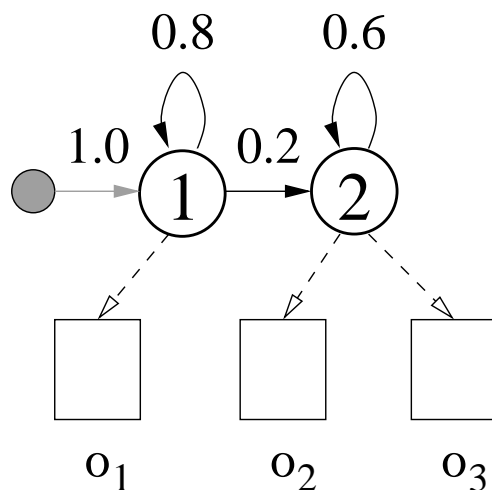
Example of the Hidden Markov Model, λ

- (a) Initial-state probabilities,
 $\pi = \{\pi_i\} = \{P(x_1 = i)\}$ for $1 \leq i \leq N$;
- (b) State-transition probabilities,
 $A = \{a_{ij}\} = \{P(x_t = j | x_{t-1} = i)\}$ for $1 \leq i, j \leq N$;
- (c) Discrete output probabilities,
 $B = \{b_i(k)\} = \{P(o_t = k | x_t = i)\}$ for $1 \leq i \leq N$
and $1 \leq k \leq K$.



producing observations
 $\mathcal{O} = \{o_1, o_2, \dots, o_6\}$
from a state sequence
 $X = \{1, 1, 2, 3, 3, 4\}$.

Probability of HMM state sequence



Output probabilities: $B = \begin{bmatrix} b_1(k) \\ b_2(k) \end{bmatrix} = \begin{bmatrix} 0.5 & 0.2 & 0.3 \\ 0 & 0.9 & 0.1 \end{bmatrix}$.

Probability with a certain state sequence, $X = \{1, 2, 2\}$:

$$\begin{aligned} P(\mathcal{O}, X|\lambda) &= P(\mathcal{O}|X, \lambda)P(X|\lambda) \\ &= \pi_1 b_1(o_1) a_{12} b_2(o_2) a_{22} b_2(o_3) \\ &= 1 \times \\ &= \\ &\approx \end{aligned}$$

(24)

Three tasks within HMM framework

1. Compute likelihood of a set of observations with a given model, $P(\mathcal{O}|\lambda)$
2. Decode a test sequence by calculating the most likely path, X^*
3. Optimise the template patterns by training the parameters in the models, $\Lambda = \{\lambda\}$.

Task 1: Computing $P(\mathcal{O}|\lambda)$

So far, we calculated the joint probability of observations and state sequence, for a given model λ :

$$P(\mathcal{O}, X|\lambda) = P(\mathcal{O}|X, \lambda)P(X|\lambda).$$

For the total probability of the observations, we marginalise the state sequence by summing over all possible X :

$$P(\mathcal{O}|\lambda) = \sum_{\text{all } X} P(\mathcal{O}, X|\lambda) = \sum_{\text{all } \mathbf{x}_1^T} P(\mathbf{x}_1^T, \mathbf{o}_1^T|\lambda). \quad (25)$$

Now, we define *forward likelihood* for state j as

$$\alpha_t(j) = P(\mathbf{x}_1^t, \mathbf{o}_1^t|\lambda). \quad (26)$$

and apply the HMM's simplifying assumptions to yield

$$\alpha_t(j) = \sum_{i=1}^N \alpha_{t-1}(i) P(x_t = j|x_{t-1} = i, \lambda) P(o_t|x_t = j, \lambda), \quad (27)$$

where the probability of current state x_t depends only on the previous state x_{t-1} , and the current observation o_t depends only on current state (Gold & Morgan, 2000).

Forward procedure

To calculate *forward likelihood*, $\alpha_t(i) = P(x_t = i, \mathbf{o}_1^t | \lambda)$:

1. Initialise,

$$\alpha_1(i) = \pi_i b_i(o_1), \quad \text{for } 1 \leq i \leq N;$$

2. Recur for $t = 2, 3, \dots, T$,

$$\alpha_t(j) = \left[\sum_{i=1}^N \alpha_{t-1}(i) a_{ij} \right] b_j(o_t), \quad \text{for } 1 \leq j \leq N;$$

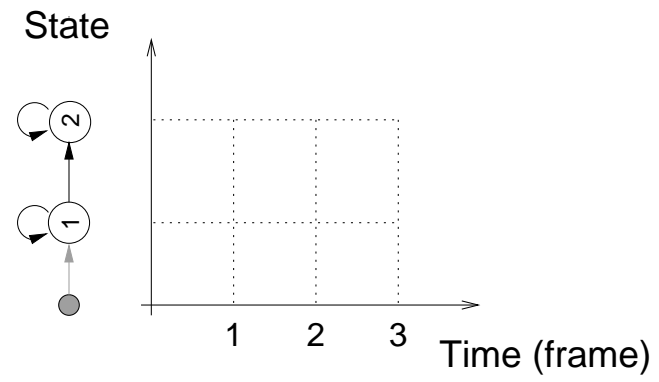
(28)

3. Finalise,

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

Thus, we can solve Task 1 efficiently by recursion.

Worked example of the forward procedure



Backward procedure

We define *backward likelihood*, $\beta_t(i) = P(\mathbf{o}_{t+1}^T | x_t = i, \lambda)$, and calculate:

1. Initialise,

$$\beta_T(i) = 1, \quad \text{for } 1 \leq i \leq N;$$

2. Recur for $t = T - 1, T - 2, \dots, 1$,

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \quad \text{for } 1 \leq i \leq N;$$

(29)

3. Finalise,

$$P(\mathcal{O} | \lambda) = \sum_{i=1}^N \pi_i b_i(o_1) \beta_1(i).$$

This is an equivalent way of computing $P(\mathcal{O} | \lambda)$ efficiently by recursion.

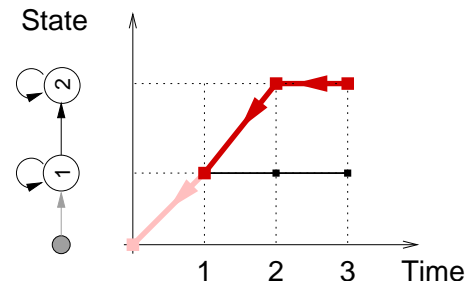
Part 1 summary

- Distance-from-template pattern matching by DTW:
 - Problems: templates, distance metric, penalties
 - Solution: probabilistic approach
- Probability fundamentals:
 - Normalisation and marginalisation
 - Joint and conditional probabilities, Bayes theorem
- Markov models:
 - sequence of directly observable states
- Hidden Markov models (HMMs):
 - hidden state sequence
 - generation of observations
- Computing α_t , β_t and $P(\mathcal{O}|\lambda)$:
 - by forward procedure
 - by backward procedure

HMM part 2

Dr Philip Jackson

- Task 2:
how to find the best path:
 - Viterbi algorithm
- Task 3:
how to train the models:
 - Viterbi training
 - Expectation maximisation
 - Baum-Welch formulae



Task 2: finding the best path

Given observations $\mathcal{O} = \{o_1, \dots, o_T\}$, find the HMM state sequence $X = \{x_1, \dots, x_T\}$ that has greatest likelihood

$$X^* = \arg \max_X P(\mathcal{O}, X|\lambda), \quad (30)$$

where

$$\begin{aligned} P(\mathcal{O}, X|\lambda) &= P(\mathcal{O}|X, \lambda)P(X|\lambda) \\ &= \pi_{x_1} b_{x_1}(o_1) \cdot \prod_{t=2}^T a_{x_{t-1}x_t} b_{x_t}(o_t). \end{aligned} \quad (31)$$

The *Viterbi algorithm* is an inductive method for finding the optimal state sequence X^* efficiently.

Task 2: Viterbi algorithm

1. Initialise,

$$\delta_1(i) = \pi_i b_i(o_1)$$

$$\psi_1(i) = 0$$

for $1 \leq i \leq N$;

2. Recur for $t = 2, \dots, T$,

$$\delta_t(j) = \max_i [\delta_{t-1}(i) a_{ij}] b_j(o_t)$$

$$\psi_t(j) = \arg \max_i [\delta_{t-1}(i) a_{ij}]$$

for $1 \leq j \leq N$;

3. Finalise,

$$P(\mathcal{O}, X^* | \lambda) = \max_i [\delta_T(i)]$$

$$x_T^* = \arg \max_i [\delta_T(i)]$$

4. Trace back, for $t = T - 1, T - 2, \dots, 1$,

$$x_t^* = \psi_{t+1}(x_{t+1}^*), \text{ and } X^* = \{x_1^*, x_2^*, \dots, x_T^*\}.$$

(32)

Illustration of the Viterbi algorithm

1. Initialise,

$$\delta_1(i) = \pi_i b_i(o_1)$$

$$\psi_1(i) = 0$$

2. Recur for $t = 2$,

$$\delta_2(j) = \max_i [\delta_1(i) a_{ij}] b_j(o_2)$$

$$\psi_2(j) = \arg \max_i [\delta_1(i) a_{ij}]$$

Recur for $t = 3$,

$$\delta_3(j) = \max_i [\delta_2(i) a_{ij}] b_j(o_3)$$

$$\psi_3(j) = \arg \max_i [\delta_2(i) a_{ij}]$$

3. Finalise,

$$P(\mathcal{O}, X^* | \lambda) = \max_i [\delta_3(i)]$$

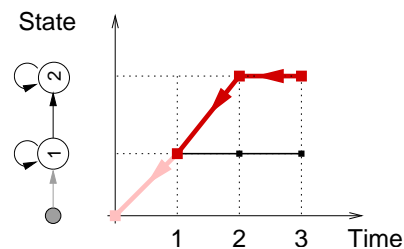
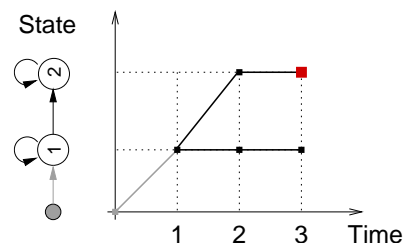
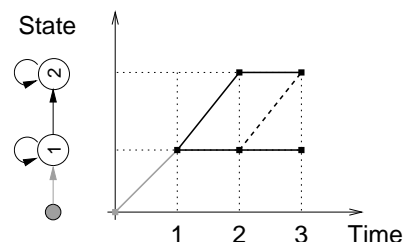
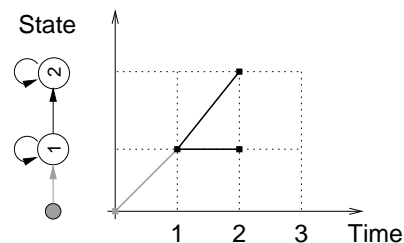
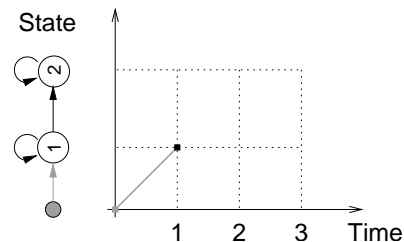
$$x_3^* = \arg \max_i [\delta_3(i)]$$

4. Trace back for $t=2..1$,

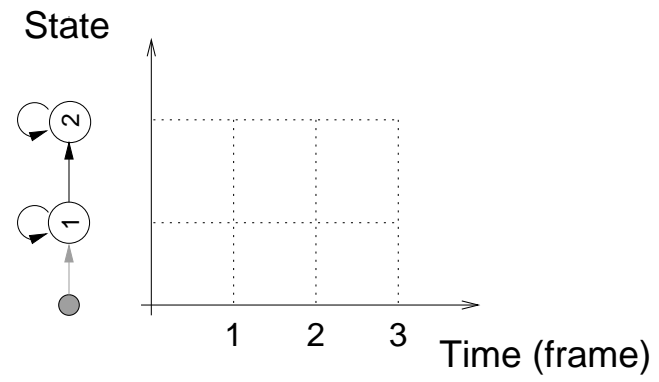
$$x_2^* = \psi_3(x_3^*)$$

$$x_1^* = \psi_2(x_2^*)$$

$$X^* = \{x_1^*, x_2^*, x_3^*\}.$$



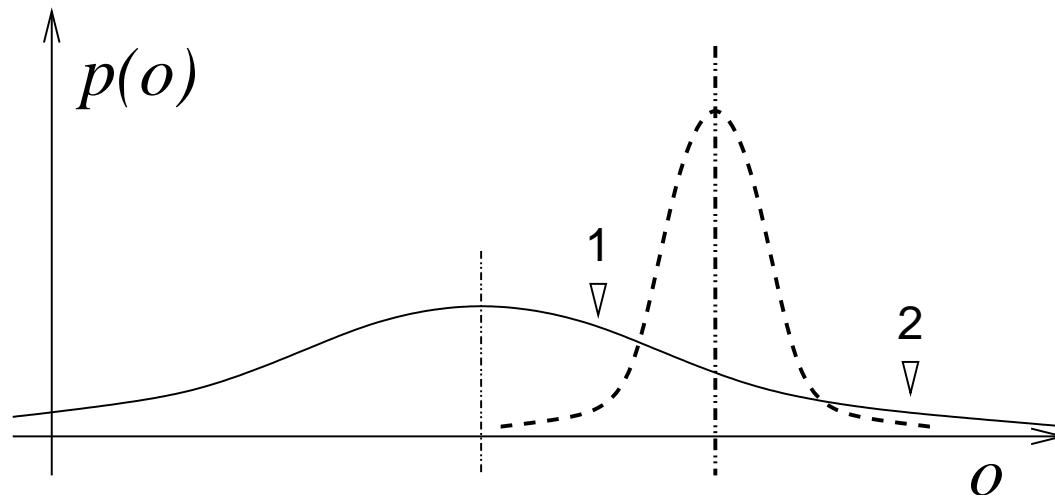
Worked example of Viterbi algorithm



Task 3: training the models

Motivation for the most likely model parameters

Given two different probability distribution functions (pdfs),



consider how data in regions 1 and 2 would be treated by a least-squares approach, and compare this to their relative likelihoods.

Maximum likelihood training

In general, we want to find the value of some model parameter κ that is most likely to give our set of training data $\mathcal{O}_{\text{train}}$.

This maximum likelihood (ML) estimate $\hat{\kappa}$ is obtained by setting the derivative of $P(\mathcal{O}_{\text{train}}|\kappa)$ wrt. κ equal to zero, which is equivalent to:

$$\frac{\partial \ln P(\mathcal{O}_{\text{train}}|\kappa)}{\partial \kappa} = 0, \quad (33)$$

This *likelihood equation* enables us to optimise the model parameters in training.

Re-estimating the model parameters λ

As a preliminary approach, let us use the optimal path X^* computed by the Viterbi algorithm with some initial model parameters $\lambda = \{\pi, A, B\}$. In so doing, we approximate the total likelihood of the observations:

$$P(\mathcal{O}, X|\lambda) = \sum_{\text{all } X} P(\mathcal{O}, X|\lambda) \approx P(\mathcal{O}, X^*|\lambda). \quad (34)$$

Using X^* , we can make a hard binary decision about the occupation of state i , $q_t(i) \in \{0, 1\}$, and train the parameters of our model accordingly.

Viterbi training (hard state assignment)

Model parameters can be re-estimated using the Viterbi state alignment (aka. segmental k-means training):

(a) Initial-state probabilities,

$$\hat{\pi}_i = q_1(i) \quad \text{for } 1 \leq i \leq N;$$

$$\text{where state indicator } q_t(i) = \begin{cases} 1 & \text{for } i = x_t; \\ 0 & \text{otherwise.} \end{cases}$$

(b) State-transition probabilities,

$$\hat{a}_{ij} = \frac{\sum_{t=2}^T q_{t-1}(i)q_t(j)}{\sum_{t=2}^T q_{t-1}(i)} \quad \text{for } 1 \leq i, j \leq N;$$

(c) Discrete output probabilities,

$$\hat{b}_j(k) = \frac{\sum_{t=1}^T \omega_t(k)q_t(j)}{\sum_{t=1}^T q_t(j)} \quad \text{for } 1 \leq j \leq N;$$

and $1 \leq k \leq K$.

$$\text{where event indicator } \omega_t(k) = \begin{cases} 1 & \text{for } k = o_t; \\ 0 & \text{otherwise.} \end{cases}$$

Maximum likelihood training by EM

Baum-Welch re-estimation (occupation)

However, the hidden state occupation is not known with absolute certainty. So, the method of expectation maximisation (EM) is used to optimise the model parameters based on the *occupation likelihood*,

$$\gamma_t(i) = P(x_t = i | \mathcal{O}, \lambda), \quad (35)$$

which we can rearrange using Bayes theorem:

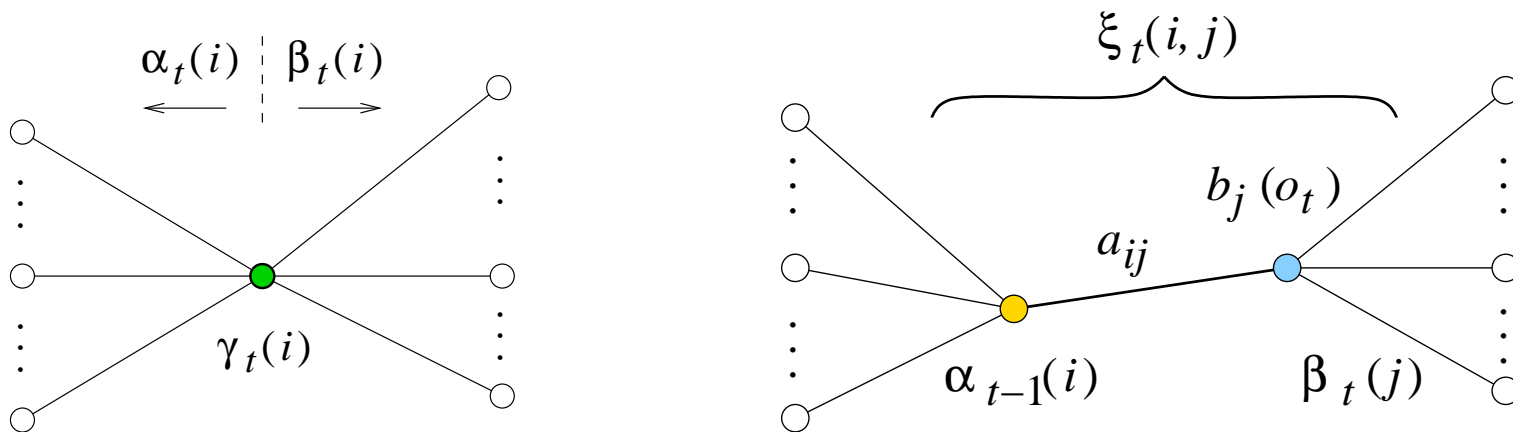
$$\begin{aligned} \gamma_t(i) &= \frac{P(x_t = i, \mathcal{O} | \lambda)}{P(\mathcal{O} | \lambda)} \\ &= \frac{\alpha_t(i) \beta_t(i)}{P(\mathcal{O} | \lambda)}, \end{aligned} \quad (36)$$

where α_t and β_t are computed by the forward and backward procedures, which yield $P(\mathcal{O} | \lambda)$.

Baum-Welch re-estimation (transition)

We also define a *transition likelihood*,

$$\begin{aligned}
 \xi_t(i, j) &= P(x_{t-1} = i, x_t = j | \mathcal{O}, \lambda) \\
 &= \frac{P(x_{t-1} = i, x_t = j, \mathcal{O} | \lambda)}{P(\mathcal{O} | \lambda)} \\
 &= \frac{\alpha_{t-1}(i) a_{ij} b_j(o_t) \beta_t(j)}{P(\mathcal{O} | \lambda)}. \tag{37}
 \end{aligned}$$



Trellis fragments depicting (left) occupation and (right) transition likelihood calculation.

Baum-Welch training (soft state assignment)

(a) Initial-state probabilities,

$$\hat{\pi}_i = \gamma_1(i) \quad \text{for } 1 \leq i \leq N;$$

(b) State-transition probabilities,

$$\hat{a}_{ij} = \frac{\sum_{t=2}^T \xi_t(i,j)}{\sum_{t=2}^T \gamma_{t-1}(i)} \quad \text{for } 1 \leq i, j \leq N;$$

(c) Discrete output probabilities,

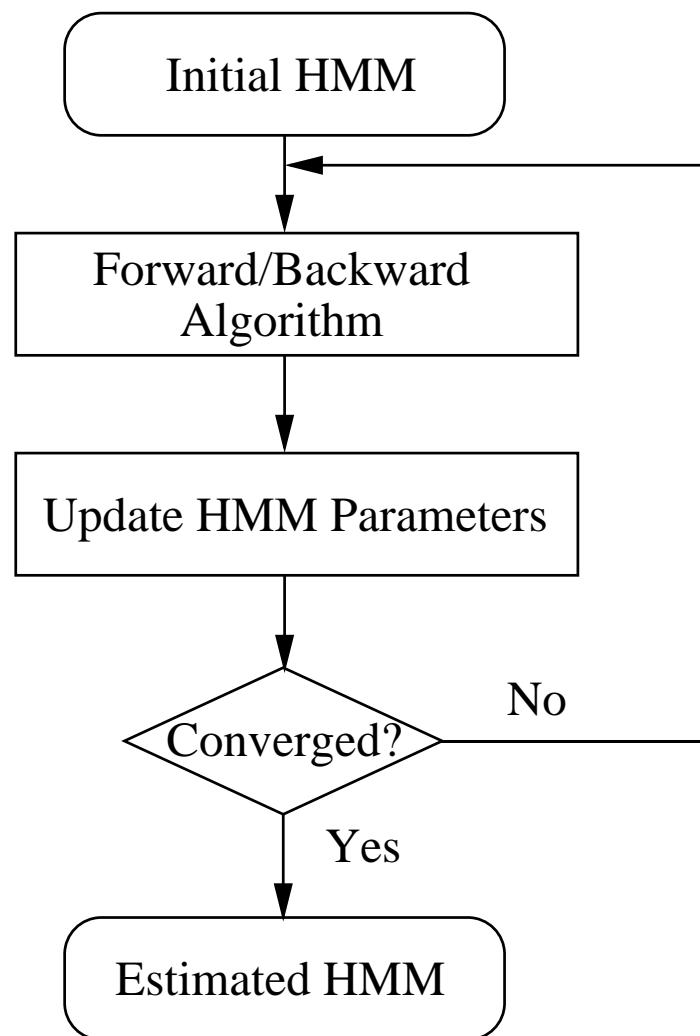
$$\hat{b}_j(k) = \frac{\sum_{t=1}^T \mathbb{1}_{o_t=k} \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)} \quad \begin{array}{l} \text{for } 1 \leq j \leq N; \\ \text{and } 1 \leq k \leq K. \end{array}$$

It can be shown that re-estimation increases the likelihood of the training data for the new model $\hat{\lambda}$,

$$P(\mathcal{O}_{\text{train}}|\hat{\lambda}) \geq P(\mathcal{O}_{\text{train}}|\lambda),$$

although it does not guarantee a global maximum.

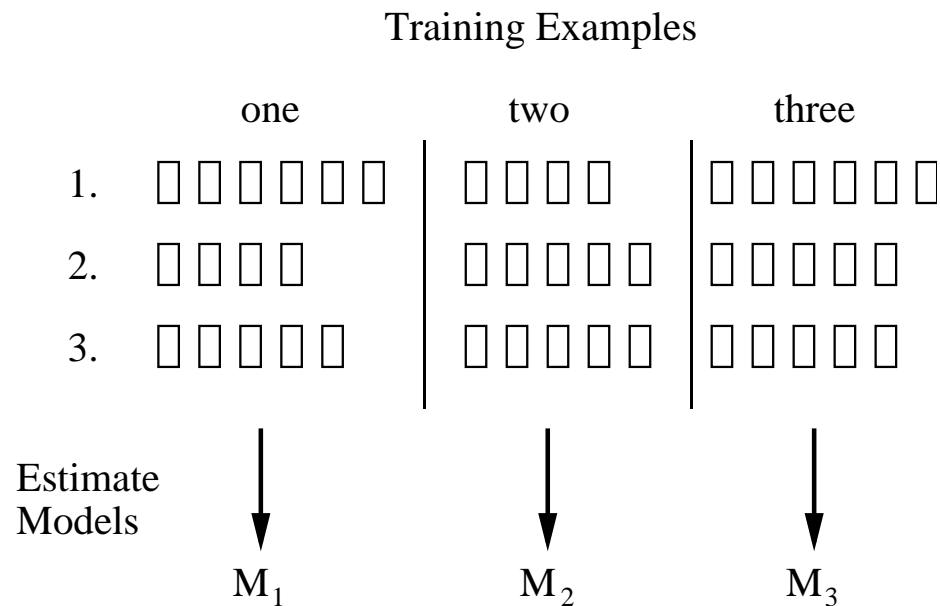
Overview of the training procedure



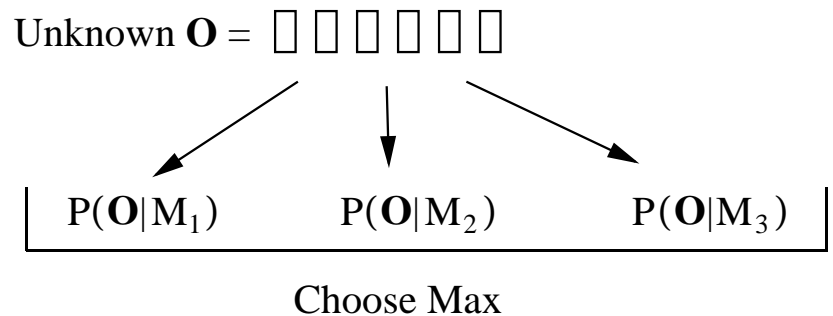
Parameter re-estimation based on a single training example (Young et al., 2002).

Use of HMMs with training and test data

(a) Training



(b) Recognition



Training and isolated-word recognition (Young et al., 2002).

Part 2 summary

- Viterbi algorithm to find the best state sequence X^*
- Re-estimation of model parameters, $\lambda = \{\pi, A, B\}$:
 - Viterbi training
 - Occupation and transition likelihoods
 - Baum-Welch training

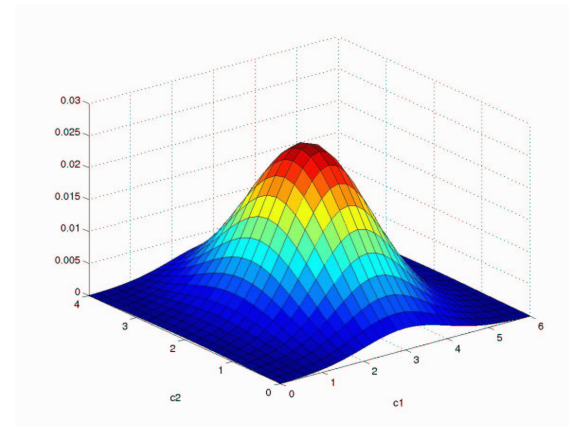
Homework

- Perform Viterbi training on worked example
- Calculate γ_t and ξ_t likelihoods
- Perform Baum-Welch training, and compare

HMM part 3

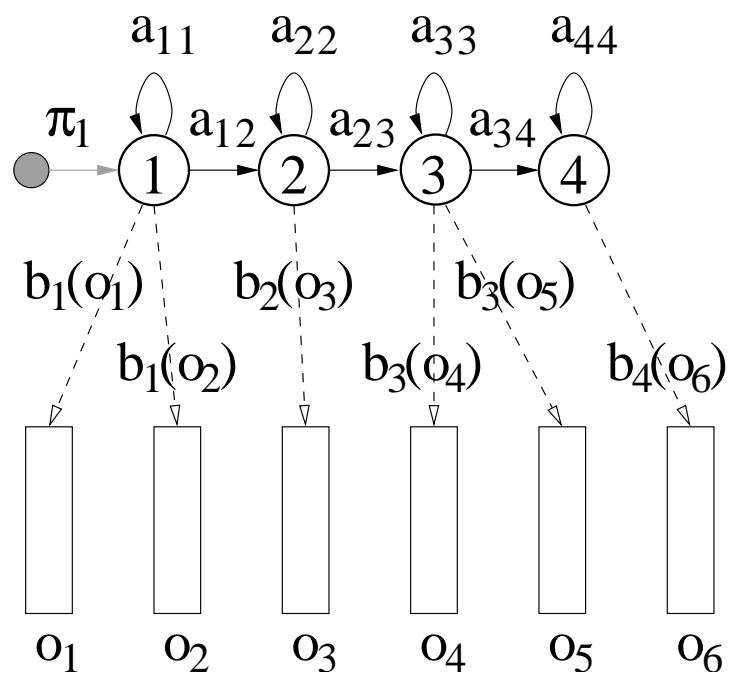
Dr Philip Jackson

- Discrete & continuous HMMs
- Gaussian output pdfs
 - Univariate Gaussian
 - Multivariate Gaussian
 - Mixtures of Gaussians
- Practical issues
- Summary



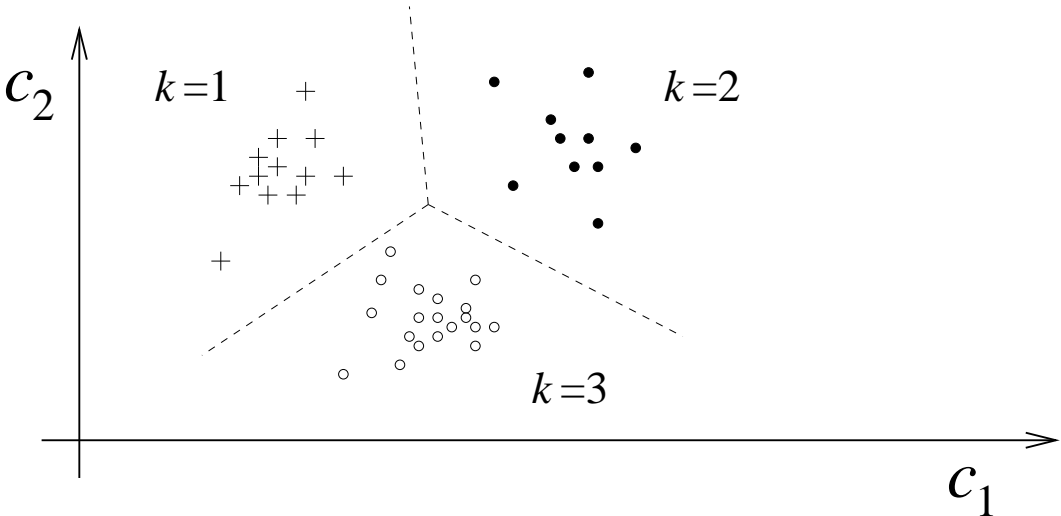
Discrete HMM, λ

- (a) Initial-state probabilities,
 $\pi = \{\pi_i\} = \{P(x_1 = i)\}$ for $1 \leq i \leq N$;
- (b) State-transition probabilities,
 $A = \{a_{ij}\} = \{P(x_t = j | x_{t-1} = i)\}$ for $1 \leq i, j \leq N$;
- (c) Discrete output probabilities,
 $B = \{b_i(k)\} = \{P(o_t = k | x_t = i)\}$ for $1 \leq i \leq N$
and $1 \leq k \leq K$.

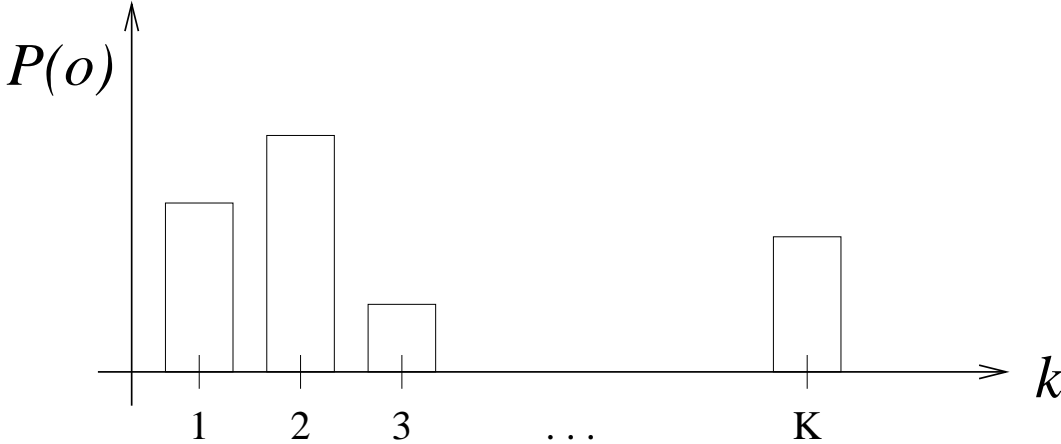


producing
discrete observations
 $\mathcal{O} = \{o_1, o_2, \dots, o_6\}$
with state sequence
 $X = \{1, 1, 2, 3, 3, 4\}$.

Observations in discretised feature space



Discrete output probability histogram



Continuous HMM, λ

- (a) Initial-state probabilities,
 $\pi = \{\pi_i\} = \{P(x_1 = i)\}$ for $1 \leq i \leq N$;
- (b) State-transition probabilities,
 $A = \{a_{ij}\} = \{P(x_t = j | x_{t-1} = i)\}$ for $1 \leq i, j \leq N$;
- (c) Continuous output probability densities,
 $B = \{b_i(o_t)\} = \{p(o_t | x_t = i)\}$ for $1 \leq i \leq N$,

where the output pdf for each state i can be Gaussian,

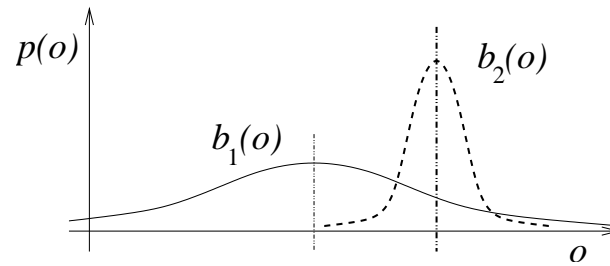
$$\begin{aligned} b_i(o_t) &= \mathcal{N}(o_t; \mu_i, \Sigma_i) \\ &= \frac{1}{\sqrt{2\pi\Sigma_i}} \exp\left(-\frac{(o - \mu_i)^2}{2\Sigma_i}\right), \end{aligned} \quad (38)$$

and $\mathcal{N}(\cdot)$ is a normal distribution with mean μ_i and variance Σ_i , evaluated at o_t .

Univariate Gaussian (scalar observations)

For a given state i ,

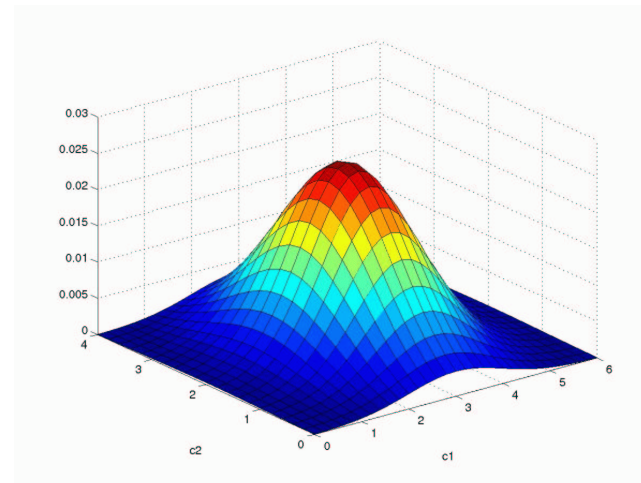
$$b_i(o_t) = \frac{1}{\sqrt{2\pi\Sigma_i}} \exp\left[-\frac{(o_t - \mu_i)^2}{2\Sigma_i}\right].$$



Multivariate Gaussian (vector observations)

$$b_i(\mathbf{o}_t) = \frac{1}{\sqrt{(2\pi)^K |\Sigma_i|}} \exp\left[-\frac{1}{2}(\mathbf{o}_t - \boldsymbol{\mu}_i)\Sigma_i^{-1}(\mathbf{o}_t - \boldsymbol{\mu}_i)'\right],$$

where the dimensionality of the observation space is K .



Baum-Welch training of Gaussian state parameters

For observations produced by an HMM with a continuous multivariate Gaussian distribution, i.e.:

$$b_i(\mathbf{o}_t) = \mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \quad (39)$$

we can again make a soft (i.e., probabilistic) allocation of the observations to the states. Thus, if $\gamma_t(i)$ denotes the likelihood of occupying state i at time t then the ML estimates of the Gaussian output pdf parameters become weighted averages,

$$\hat{\boldsymbol{\mu}}_i = \frac{\sum_{t=1}^T \gamma_t(i) \mathbf{o}_t}{\sum_{t=1}^T \gamma_t(i)} \quad (40)$$

and

$$\hat{\boldsymbol{\Sigma}}_i = \frac{\sum_{t=1}^T \gamma_t(i) (\mathbf{o}_t - \boldsymbol{\mu}_i) (\mathbf{o}_t - \boldsymbol{\mu}_i)',}{\sum_{t=1}^T \gamma_t(i)}, \quad (41)$$

normalised by a denominator which is the total likelihood of all paths passing through state i .

Gaussian mixture pdfs

Univariate Gaussian mixture

$$\begin{aligned} b_i(o_t) &= \sum_{m=1}^M c_{im} \mathcal{N}(o_t; \mu_{im}, \Sigma_{im}) \\ &= \sum_{m=1}^M \frac{c_{im}}{\sqrt{2\pi\Sigma_{im}}} \exp\left[-\frac{(o_t - \mu_{im})^2}{2\Sigma_{im}}\right], \end{aligned} \quad (42)$$

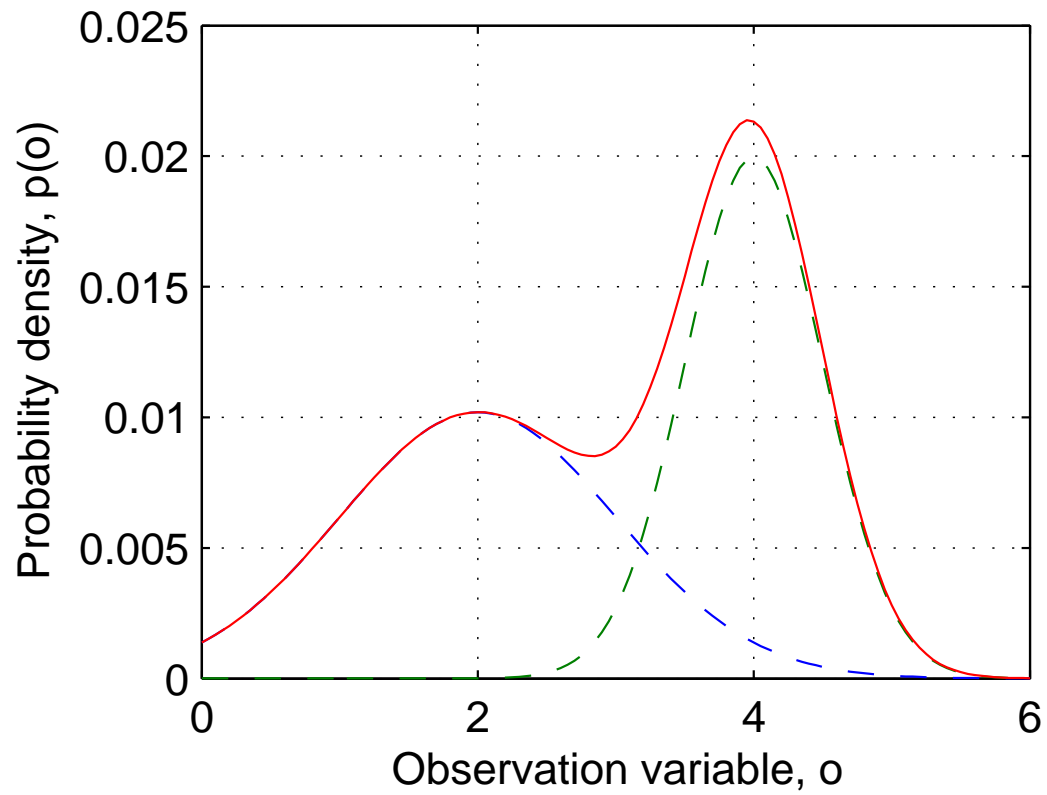
where M is the number of mixture components (M -mix), and the mixture weights sum to one: $\sum_{m=1}^M c_{im} = 1$.

Multivariate Gaussian mixture

$$b_i(\mathbf{o}_t) = \sum_{m=1}^M c_{im} \mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_{im}, \boldsymbol{\Sigma}_{im}), \quad (43)$$

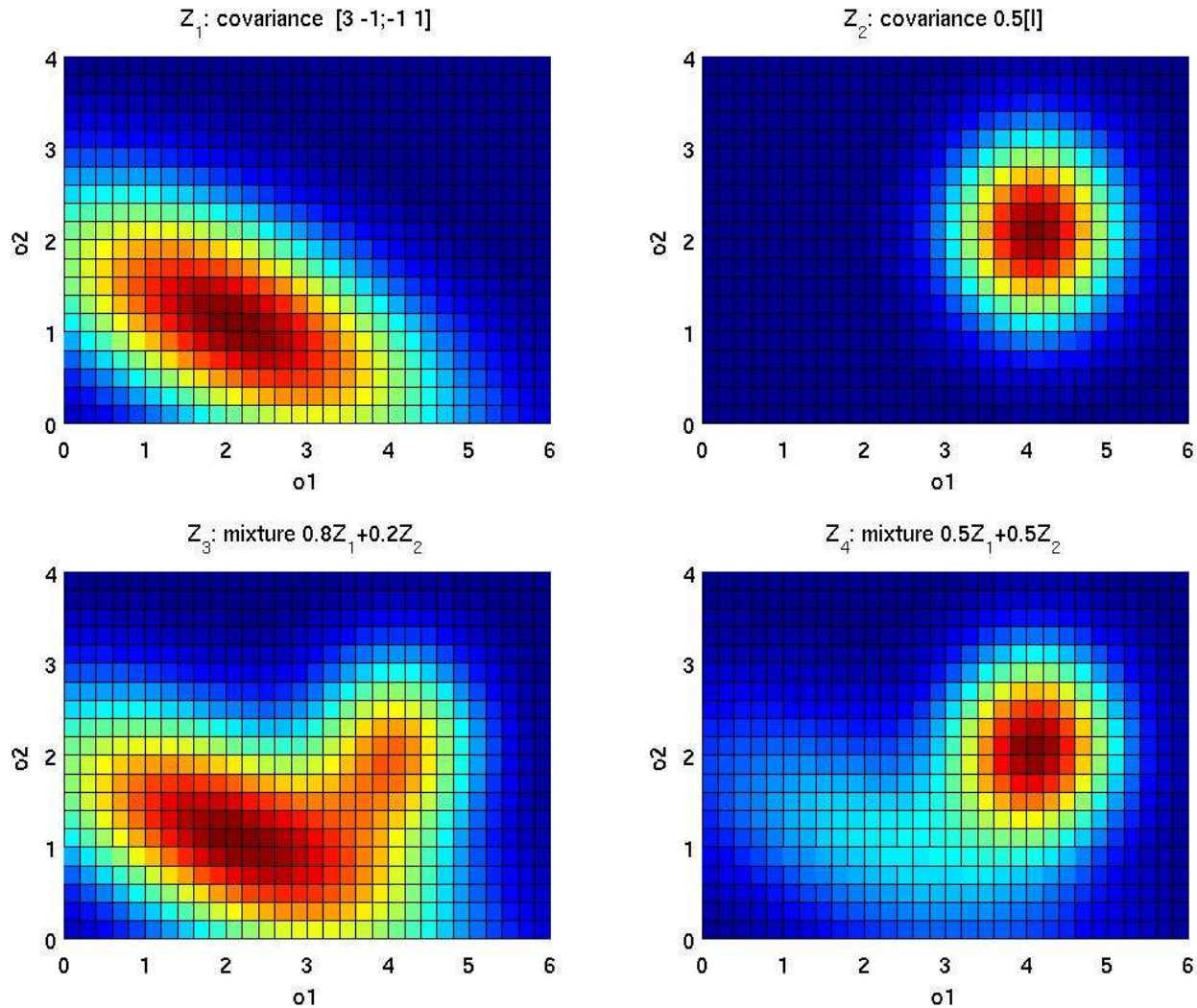
where $\mathcal{N}(\cdot)$ is the multivariate normal distribution with vector mean $\boldsymbol{\mu}_{im}$ and covariance matrix $\boldsymbol{\Sigma}_{im}$, evaluated at \mathbf{o}_t .

Univariate mixture of Gaussians



Mixture of two univariate Gaussian components.

Multivariate mixtures of Gaussians



Examples of two multivariate Gaussian pdfs (upper), and two Gaussian mixtures of them with different weights (lower).

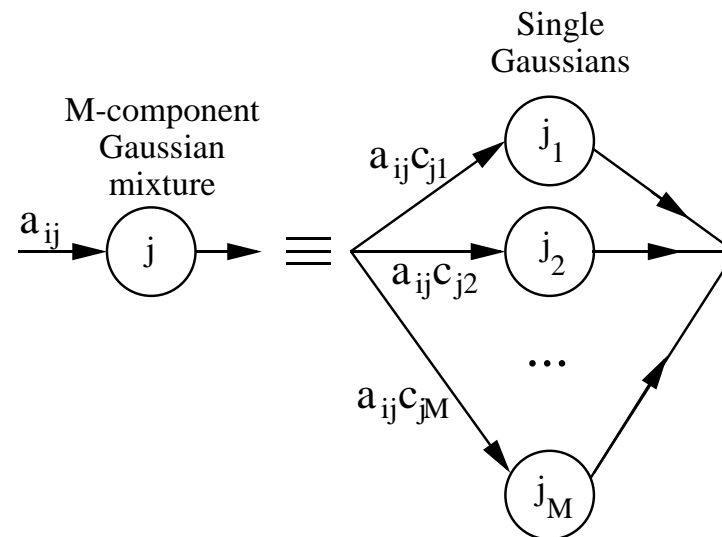
Baum-Welch training with mixtures

We define the *mixture-occupation likelihood*:

$$\gamma_t(j, m) = \frac{\alpha_t^-(j) c_{jm} \mathcal{N}(\mathbf{o}_t; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}) \beta_t(j)}{P(\mathcal{O}|\lambda)}, \quad (44)$$

where $\gamma_t(j) = \sum_{m=1}^M \gamma_t(j, m)$,

and $\alpha_t^-(j) = \begin{cases} \pi_j & \text{for } t=1, \\ \sum_{i=1}^N \alpha_{t-1}(i) a_{ij} & \text{otherwise.} \end{cases}$



Representing occupation of a mixture (Young et al., 2002).

Baum-Welch re-estimation of mixture parameters

Using soft assignment of observations to mixture components given by mixture-occupation likelihood $\gamma_t(j, m)$, we train our parameters with revised update equations.

Mean vector:

$$\hat{\boldsymbol{\mu}}_{jm} = \frac{\sum_{t=1}^T \gamma_t(j, m) \mathbf{o}_t}{\sum_{t=1}^T \gamma_t(j, m)}, \quad (45)$$

Covariance matrix:

$$\hat{\boldsymbol{\Sigma}}_{jm} = \frac{\sum_{t=1}^T \gamma_t(j, m) (\mathbf{o}_t - \boldsymbol{\mu}_{jm})(\mathbf{o}_t - \boldsymbol{\mu}_{jm})'}{\sum_{t=1}^T \gamma_t(j, m)}, \quad (46)$$

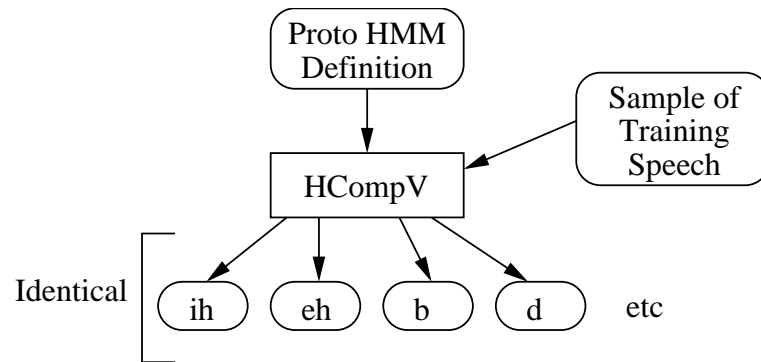
Mixture weights:

$$\hat{c}_{jm} = \frac{\sum_{t=1}^T \gamma_t(j, m)}{\sum_{t=1}^T \gamma_t(j)}. \quad (47)$$

Practical issues

Model initialisation

1. Random
2. Flat start
3. Viterbi alignment
(supervised/
unsupervised)

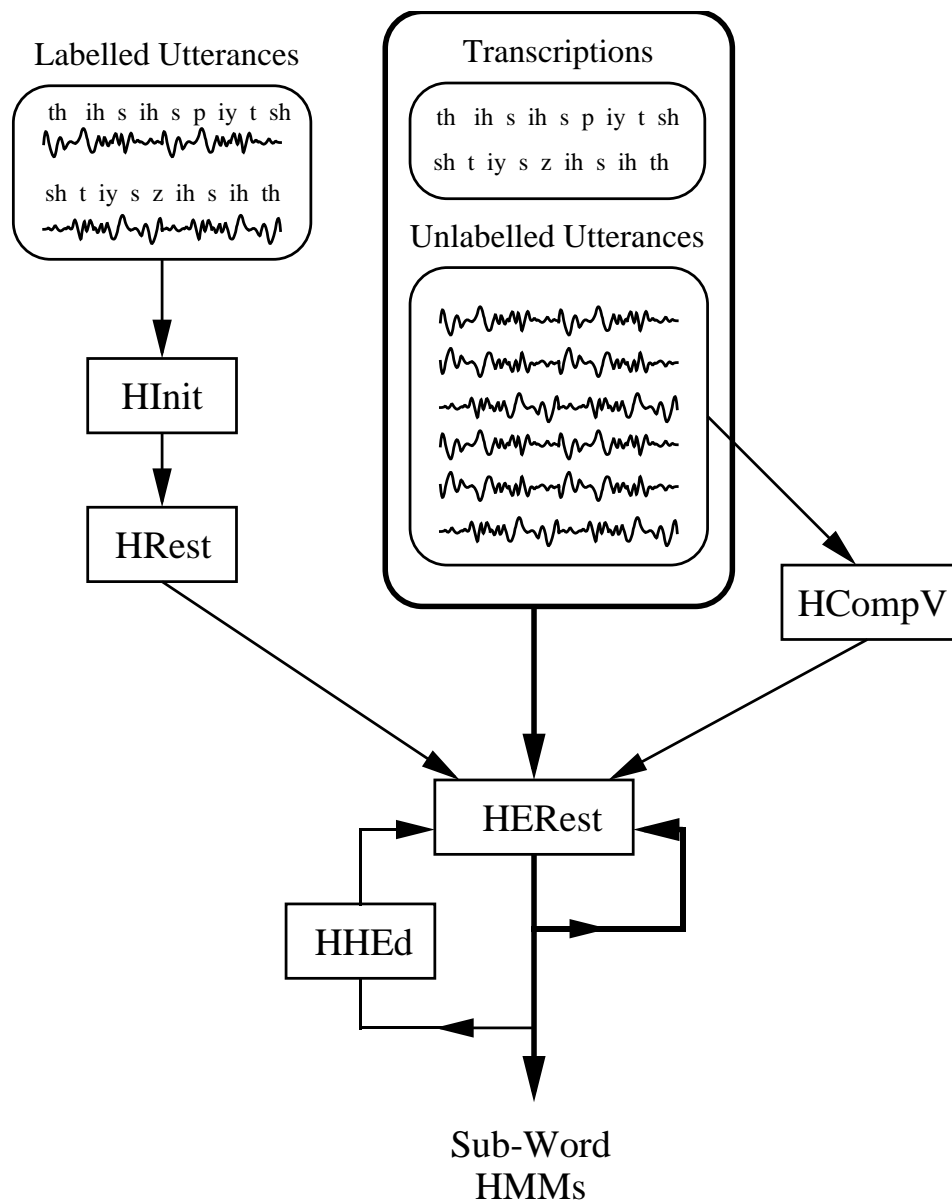


Flat start (Young et al., 2002).

Decoding

- Probabilities stored as log probs to avoid underflow
- Paths propagated through trellis by token passing
- Search space kept tractable by beam pruning

Re-estimation and embedded re-estimation



HMM training with variously labelled data (Young et al., 2002).

Number of parameters & Regularisation

- Context sensitivity
- Parsimonious models
 - Occam's razor
- Size of database
- variance floor
- parameter tying
 - agglomerative clustering
 - decision trees

Related topics that are not covered here

- Noise robustness
 - factorial HMMs
- Adaptation
 - MLLR & MAP
- Language modeling
- Markov random fields
- Monte Carlo Markov chains

Part 3 summary

- Introduction to continuous HMMs
- Gaussian output pdfs
 - Univariate & multivariate Gaussians
 - Mixtures of Gaussians
- Baum-Welch training of continuous HMMs
- Practical issues

References

- B. Gold & N. Morgan, *Speech and Audio Signal Processing*, New York: Wiley, pp.346–347, 2000 [0-471-35154-7].
- J. N. Holmes & W. J. Holmes, *Speech Synthesis and Recognition*, Taylor & Francis, 2001 [0-748-40857-6].
- F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, 1998 [0-262-10066-5].
- D. Jurafsky & J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*, Prentice Hall, 2003 [0-131-22798-X].
- L. R. Rabiner. *A tutorial on HMM and selected applications in speech recognition*. In *Proc. IEEE*, Vol. 77, No. 2, pp. 257–286, 1989.
- S. J. Young, et al., *The HTK Book*, Cambridge Univ. Eng. Dept. (v3.2), 2002 [<http://htk.eng.cam.ac.uk/>].