CHAPTER 23

Discrete-State Markov Models

23.1 Markov Models

Time-series are datasets for which the constituent datapoints can be naturally ordered. This order often corresponds to an underlying single physical dimension, typically time, though any other single dimension may be used. The time-series models we consider are probability models over a collection of random variables $v_1, \ldots, v_T$ with individual variables v_t indexed by a time index t. These indices are elements of the index set $\mathcal{T}$. For nonnegative indices, $\mathcal{T} = \mathbb{N}^+$, the model is a discrete-time process. Continuous-time processes, $\mathcal{T} = \mathbb{R}$, are natural in particular application domains yet require additional notation and concepts. We therefore focus exclusively on discrete-time models. A probabilistic time-series model requires a specification of the joint distribution $p(v_1, \ldots, v_T)$. For the case in which the observed data v_t is discrete, the joint probability table for $p(v_1, \ldots, v_T)$ has exponentially many entries. We cannot expect to independently specify all the exponentially many entries and are therefore forced to make simplified models under which these entries can be parameterised in a lower dimensional manner. Such simplifications are at the heart of time-series modelling and we will discuss some classical models in the following sections.

Definition 107 (Time-Series Notation)**.**

$$x_{a:b} \equiv x_a, x_{a+1}, \ldots, x_b, \qquad \text{with } x_{a:b} = x_a \text{ for } b \leq a \tag{23.1.1}$$

For timeseries data $v_1, \ldots, v_T$, we need a model $p(v_{1:T})$. For consistency with the causal nature of time, it is meaningful to consider the decomposition

$$p(v_{1:T}) = \prod_{t=1}^{T} p(v_t | v_{1:t-1}) \tag{23.1.2}$$

with the convention $p(v_t|v_{1:t-1}) = p(v_1)$ for $t = 1$. It is often natural to assume that the influence of the immediate past is more relevant than the remote past and in Markov models only a limited number of previous observations are required to predict the future.

Definition 108 (Markov chain)**.** A Markov chain defined on either discrete or continuous variables $v_{1:T}$ is one in which the following conditional independence assumption holds:

$$p(v_t | v_1, \ldots, v_{t-1}) = p(v_t | v_{t-L}, \ldots, v_{t-1}) \tag{23.1.3}$$

Figure 23.1: **(a)**: First order Markov chain. **(b)**: Second order Markov chain.

where $L \geq 1$ is the *order* of the Markov chain and $v_t = \emptyset$ for $t < 1$. For a first order Markov chain,

$$p(v_{1:T}) = p(v_1)p(v_2|v_1)p(v_3|v_2)\dots p(v_T|v_{T-1}) \tag{23.1.4}$$

For a *stationary Markov chain* the transitions $p(v_t = s'|v_{t-1} = s) = f(s', s)$ are time-independent. Otherwise the chain is non-stationary, $p(v_t = s'|v_{t-1} = s) = f(s', s, t)$.

23.1.1 Equilibrium and stationary distribution of a Markov chain

The stationary distribution p_∞ of a Markov chain with transition matrix $\mathbf{M}$ is defined by the condition

$$p_\infty(i) = \sum_j \underbrace{p(x_t = i|x_{t-1} = j)}_{M_{ij}} p_\infty(j) \tag{23.1.5}$$

In matrix notation this can be written as the vector equation

$$\mathbf{p}_\infty = \mathbf{M}\mathbf{p}_\infty \tag{23.1.6}$$

so that the stationary distribution is proportional to the eigenvector with unit eigenvalue of the transition matrix. Note that there may be more than one stationary distribution. See exercise(217) and [122].

Given a state x_1, we can iteratively draw samples $\mathsf{x}_2, \dots, \mathsf{x}_t$ from the Markov chain drawing a sample from $p(x_2|x_1 = \mathsf{x}_1)$, and then from $p(x_3|\mathsf{x}_2)$ *etc.* As we repeatedly sample a new state from the chain, the distribution at time t, for an initial distribution $\mathbf{p}_1(i) = \delta(i, \mathsf{x}_1)$, is

$$\mathbf{p}_t = \mathbf{M}^t\mathbf{p}_1 \tag{23.1.7}$$

If for $t \to \infty$, $\mathbf{p}_\infty$ is independent of the initial distribution $\mathbf{p}_1$, then $\mathbf{p}_\infty$ is called the equilibrium distribution of the chain. See exercise(218) for an example of a Markov chain which does not have an equilibrium distribution.

Example 95 (*PageRank*)**.** Despite their apparent simplicity, Markov chains have been put to interesting use in information retrieval and search-engines. Define the matrix

$$A_{ij} = \begin{cases} 1 & \text{if website } j \text{ has a hyperlink to website } i \\ 0 & \text{otherwise} \end{cases} \tag{23.1.8}$$

From this we can define a Markov transition matrix with elements

$$M_{ij} = \frac{A_{ij}}{\sum_i A_{ij}} \tag{23.1.9}$$

The equilibrium distribution of this Markov Chain has the interpretation : If follow links at random, jumping from website to website, the equilibrium distribution component $p_\infty(i)$ is the relative number of times we will visit website i. This has a natural interpretation as the 'importance' of website i; if a website

is isolated in the web, it will be visited infrequently by random hopping; if a website is linked by many others it will be visited more frequently.

A crude *search engine* works then as follows. For each website i a list of words associated with that website is collected. After doing this for all websites, one can make an 'inverse' list of which websites contain word w. When a user searches for word w, the list of websites that that word is then returned, ranked according to the importance of the site (as defined by the equilibrium distribution). This is a crude summary as how early search engines worked, `infolab.stanford.edu/~backrub/google.html`.

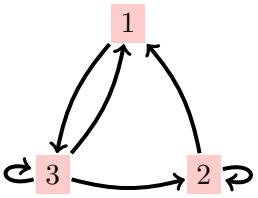


Figure 23.2: A state transition diagram for a three state Markov chain. Note that a state transition diagram is not a graphical model – it simply displays the non-zero entries of the transition matrix $p(i|j)$. The absence of link from j to i indicates that $p(i|j) = 0$.

23.1.2 Fitting Markov models

Given a sequence $v_{1:T}$, fitting a stationary Markov chain by Maximum Likelihood corresponds to setting the transitions by counting the number of observed (first-order) transitions in the sequence:

$$p(v_\tau = i|v_{\tau-1} = j) \propto \sum_{t=2}^{T} \mathbb{I}\left[v_t = i, v_{t-1} = j\right] \tag{23.1.10}$$

To show this, for convenience we write $p(v_\tau = i|v_{\tau-1} = j) \equiv \theta_{i|j}$, so that the likelihood is (assuming v_1 is known):

$$p(v_{2:T}|\theta, v_1) = \prod_{t=2}^{T}\prod_{ij} \theta_{v_t|v_{t-1}} = \prod_{t=2}^{T}\prod_{ij} \theta_{i|j}^{\mathbb{I}[v_t=i,v_{t-1}=j]} \tag{23.1.11}$$

Taking logs and adding the Lagrange constraint for the normalisation,

$$L(\theta) = \sum_{t=2}^{T}\sum_{ij} \mathbb{I}\left[v_t = i, v_{t-1} = j\right] \log \theta_{i|j} + \sum_j \lambda_j \left(1 - \sum_i \theta_{i|j}\right) \tag{23.1.12}$$

Differentiating with respect to $\theta_{i|j}$ and equating to zero, we immediately arrive at the intuitive setting, equation (23.1.10). For a set of timeseries, $v^n_{1:T_n}, n = 1, \ldots, N$, the transition is given by counting all transitions across time and datapoints. The Maximum Likelihood setting for the initial first timestep distribution is $p(v_1 = i) \propto \sum_n \mathbb{I}\left[v_1^n = i\right]$.

Bayesian fitting

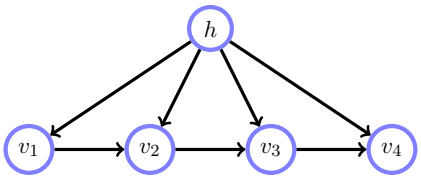
For simplicity, we assume a factorised prior on the transition

$$p(\theta) = \prod_j p(\theta_{\cdot|j}) \tag{23.1.13}$$

A convenient choice for each conditional transition is a Dirichlet distribution with hyperparameters $\mathbf{u}_j$, $p(\theta_{\cdot|j}) = \text{Dirichlet}\left(\theta_{\cdot|j}|\mathbf{u}_j\right)$, since this is conjugate to the categorical transition and

$$p(\theta|v_{1:T}) \propto p(v_{1:T}|\theta)p(\theta) \propto \prod_t \prod_{ij} \theta_{i|j}^{\mathbb{I}[v_t=i,v_{t-1}=j]} \theta_{i|j}^{u_{ij}-1} = \prod_j \text{Dirichlet}\left(\theta_{\cdot|j}|\hat{\mathbf{u}}_j\right) \tag{23.1.14}$$

where $\hat{\mathbf{u}}_j = \sum_{t=2}^{T} \mathbb{I}\left[v_{t-1} = i, v_t = j\right]$, being the number of $j \to i$ transitions in the dataset.



(a)

Figure 23.3: Mixture of first order Markov chains. The discrete hidden variable $\text{dom}(h) = \{1, \ldots, H\}$ indexes the Markov chain $p(v_t|v_{t-1}, h)$. Such models can be useful as simple sequence clustering tools.

23.1.3 Mixture of Markov models

Given a set of sequences $\mathcal{V} = \{v^n_{1:T}, n = 1, \ldots, N\}$, how might we cluster them? To keep the notation less cluttered, we assume that all sequences are of the same length T with the extension to differing lengths being straightforward. One simple approach is to fit a mixture of Markov models. Assuming the data is i.i.d., $p(\mathcal{V}) = \prod_n p(v^n_{1:T})$, we define a mixture model for a single datapoint $v_{1:T}$. Here we assume each component model is first order Markov

$$p(v_{1:T}) = \sum_{h=1}^{H} p(h)p(v_{1:T}|h) = \sum_{h=1}^{H} p(h) \prod_{t=1}^{T} p(v_t|v_{t-1}, h) \tag{23.1.15}$$

The graphical model is depicted in fig(23.3). Clustering can then be achieved by finding the Maximum Likelihood parameters $p(h)$, $p(v_t|v_{t-1}, h)$ and subsequently assigning the clusters according to $p(h|v^n_{1:T})$. Below we discuss the application of the EM algorithm to this model to learn the Maximum Likelihood parameters.

EM algorithm

Under the i.i.d. data assumption, the log likelihood is

$$\log p(\mathcal{V}) = \sum_{n=1}^{N} \log \sum_{h=1}^{H} p(h) \prod_{t=1}^{T} p(v^n_t|v^n_{t-1}, h) \tag{23.1.16}$$

For the M-step, our task is to maximise the energy

$$E = \sum_{n=1}^{N} \langle \log p(v^n_{1:T}, h) \rangle_{p^{old}(h|v^n_{1:T})} = \sum_{n=1}^{N} \left\{ \langle \log p(h) \rangle_{p^{old}(h|v^n_{1:T})} + \sum_{t=1}^{T} \langle \log p(v_t|v_{t-1}, h) \rangle_{p^{old}(h|v^n_{1:T})} \right\}$$

The contribution to the energy from the parameter $p(h)$ is

$$\sum_{n=1}^{N} \langle \log p(h) \rangle_{p^{old}(h|v^n_{1:T})} \tag{23.1.17}$$

By defining

$$\hat{p}^{old}(h) \propto \sum_{n=1}^{N} p^{old}(h|v^n_{1:T}) \tag{23.1.18}$$

one can view maximising (23.1.17) as equivalent to minimising

$$\text{KL}\Big(\hat{p}^{old}(h)|p(h)\Big) \tag{23.1.19}$$

so that the optimal choice from the M-step is to set $p^{new} = \hat{p}^{old}$, namely

$$p^{new}(h) \propto \sum_{n=1}^{N} p^{old}(h|v^n_{1:T}) \tag{23.1.20}$$

For those less comfortable with this argument, a direct maximisation including a Lagrange term to ensure normalisation of $p(h)$ can be used to derive the same result.

Similarly, the M-step for $p(v_t|v_{t-1}, h)$ is

$$p^{new}(v_t = i|v_{t-1} = j, h = k) \propto \sum_{n=1}^{N} p^{old}(h = k|v_{1:T}^n) \sum_{t=2}^{T} \mathbb{I}\left[v_t^n = i\right] \mathbb{I}\left[v_{t-1}^n = j\right] \tag{23.1.21}$$

The initial term $p(v_1|h)$ is updated using

$$p^{new}(v_1 = i|h = k) \propto \sum_{n=1}^{N} p^{old}(h = k|v_{1:T}^n) \mathbb{I}\left[v_1^n = i\right] \tag{23.1.22}$$

The E-step sets

$$p^{old}(h|v_{1:T}^n) \propto p(h)p(v_{1:T}^n|h) = p(h) \prod_{t=1}^{T} p(v_t^n|v_{t-1}^n, h) \tag{23.1.23}$$

Given an initialisation, the EM algorithm then iterates (23.1.20),(23.1.21), (23.1.22) and (23.1.23) until convergence.

For long sequences, explicitly computing the product of many terms may lead to numerical underflow issues. In practice it is therefore best to work with logs,

$$\log p^{old}(h|v_{1:T}^n) = \log p(h) + \sum_{t=1}^{T} \log p(v_t^n|v_{t-1}^n, h) + \mathsf{const.} \tag{23.1.24}$$

In this way any large constants common to all h can be removed and the distribution may be computed accurately. See `mixMarkov.m`.

Example 96 (Gene Clustering)**.** Consider the 20 fictitious gene sequences below presented in an arbitrarily chosen order. Each sequence consists of 20 symbols from the set $\{A, C, G, T\}$. The task is to try to cluster these sequences into two groups, based on the (perhaps biologically unrealistic) assumption that gene sequences in the same cluster follow a stationary Markov chain.

```
CATAGGCATTCTATGTGCTG   CCAGTTACGGACGCCGAAAG   TGGAACCTTAAAAAAAAAAA   GTCTCCTGCCCTCTCTGAAC
GTGCCTGGACCTGAAAAGCC   CGGCCGCGCCTCCGGGAACG   AAAGTGCTCTGAAAACTCAC    ACATGAACTACATAGTATAA
GTTGGTCAGCACACGGACTG   CCTCCCCTCCCCTTTCCTGC    CACTACGGCTACCTGGGCAA    CGGTCCGTCCGAGGCACTC
TAAGTGTCCTCTGCTCCTAA   CACCATCACCCTTGCTAAGG   AAAGAACTCCCCTCCCTGCC   CAAATGCCTCACGCGTCTCA
GCCAAGCAGGGTCTCAACTT   CATGGACTGCTCCACAAAGG   AAAAAAACGAAAAACCTAAG   GCGTAAAAAAAAGTCCTGGGT
```
(23.1.25)

A simple approach is to assume that the sequences are generated from a two-component $H = 2$ mixture of Markov models and train the model using Maximum Likelihood. The likelihood has local optima so that the procedure needs to be run several times and the solution with the highest likelihood chosen. One can then assign each of the sequences by examining $p(h = 1|v_{1:T}^n)$. If this posterior probability is greater than 0.5, we assign it to class 1, otherwise to class 2. Using this procedure, we find the following clusters:

```
CATAGGCATTCTATGTGCTG   TGGAACCTTAAAAAAAAAAA
CCAGTTACGGACGCCGAAAG   GTCTCCTGCCCTCTCTGAAC
CGGCCGCGCCTCCGGGAACG   GTGCCTGGACCTGAAAAGCC
 ACATGAACTACATAGTATAA   AAAGTGCTCTGAAAACTCAC
GTTGGTCAGCACACGGACTG   CCTCCCCTCCCCTTTCCTGC
CACTACGGCTACCTGGGCAA   TAAGTGTCCTCTGCTCCTAA
CGGTCCGTCCGAGGCACTCG   AAAGAACTCCCCTCCCTGCC
CACCATCACCCTTGCTAAGG   AAAAAAACGAAAAACCTAAG
CAAATGCCTCACGCGTCTCA   GCGTAAAAAAAAGTCCTGGGT
GCCAAGCAGGGTCTCAACTT
CATGGACTGCTCCACAAAGG
```
(23.1.26)

where sequences in the first column are assigned to cluster 1, and sequences in the second column to cluster 2. In this case the data in (23.1.25) was in fact generated by a two-component Markov mixture and the posterior assignment (23.1.26) is in agreement with the known clusters. See `demoMixMarkov.m`

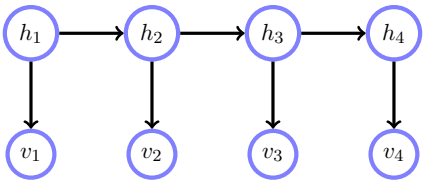


Figure 23.4: A first order hidden Markov model with 'hidden' variables $\mathrm{dom}(h_t) = \{1, \ldots, H\}$, $t = 1 : T$. The 'visible' variables v_t can be either discrete or continuous.

23.2 Hidden Markov Models

The Hidden Markov Model (HMM) defines a Markov chain on hidden (or 'latent') variables $h_{1:T}$. The observed (or 'visible') variables are dependent on the hidden variables through an emission $p(v_t|h_t)$. This defines a joint distribution

$$p(h_{1:T}, v_{1:T}) = p(v_1|h_1)p(h_1)\prod_{t=2}^{T} p(v_t|h_t)p(h_t|h_{t-1}) \tag{23.2.1}$$

for which the graphical model is depicted in fig(23.4). For a stationary HMM the transition $p(h_t|h_{t-1})$ and emission $p(v_t|h_t)$ distributions are constant through time. The use of the HMM is widespread and a subset of the many applications of HMMs is given in section(23.5).

Definition 109 (Transition Distribution)**.** For a stationary HMM the transition distribution $p(h_{t+1}|h_t)$ is defined by the $H \times H$ transition matrix

$$A_{i',i} = p(h_{t+1} = i'|h_t = i) \tag{23.2.2}$$

and an initial distribution

$$a_i = p(h_1 = i). \tag{23.2.3}$$

Definition 110 (Emission Distribution)**.** For a stationary HMM and emission distribution $p(v_t|h_t)$ with discrete states $v_t \in \{1, \ldots, V\}$, we define a $V \times H$ emission matrix

$$B_{i,j} = p(v_t = i|h_t = j) \tag{23.2.4}$$

For continuous outputs, h_t selects one of H possible output distributions $p(v_t|h_t)$, $h_t \in \{1, \ldots, H\}$.

In the engineering and machine learning communities, the term HMM typically refers to the case of discrete variables h_t, a convention that we adopt here. In statistics the term HMM often refers to any model with the independence structure in equation (23.2.1), regardless of the form of the variables h_t (see for example [54]).

23.2.1 The classical inference problems

Filtering	(Inferring the present)	$p(h_t|v_{1:t})$	
Prediction	(Inferring the future)	$p(h_t|v_{1:s})$	$t > s$
Smoothing	(Inferring the past)	$p(h_t|v_{1:u})$	$t < u$
Likelihood		$p(v_{1:T})$	
Most likely Hidden path	(Viterbi alignment)	$\underset{h_{1:T}}{\mathrm{argmax}}\, p(h_{1:T}|v_{1:T})$	

The most likely hidden path problem is termed *Viterbi alignment* in the engineering literature. All these classical inference problems are straightforward since the distribution is singly-connected, so that any standard inference method can be adopted for these problems. The factor graph and junction trees for

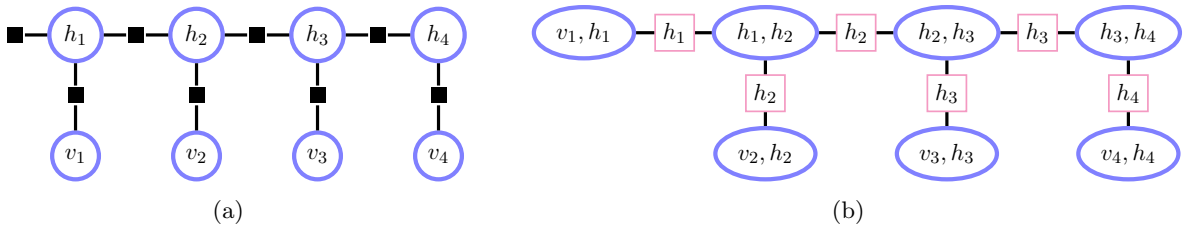


Figure 23.5: **(a)**: Factor graph for the first order HMM of fig(23.4). **(b)**: Junction tree for fig(23.4).

the first order HMM are given in fig(23.5). In both cases, after suitable setting of the factors and clique potentials, filtering corresponds to passing messages from left to right and upwards; smoothing corresponds to a valid schedule of message passing/absorption both forwards and backwards along all edges. It is also straightforward to derive appropriate recursions directly. This is instructive and also useful in constructing compact and numerically stable algorithms.

23.2.2 Filtering $p(h_t|v_{1:t})$

We first compute the joint marginal $p(h_t, v_{1:t})$ from which the conditional marginal $p(h_t|v_{1:t})$ can subsequently be obtained by normalisation. A recursion for $p(h_t, v_{1:t})$ is obtained by considering:

$$p(h_t, v_{1:t}) = \sum_{h_{t-1}} p(h_t, h_{t-1}, v_{1:t-1}, v_t) \tag{23.2.5}$$

$$= \sum_{h_{t-1}} p(v_t|\cancel{v_{1:t-1}}, h_t, \cancel{h_{t-1}})p(h_t|\cancel{v_{1:t-1}}, h_{t-1})p(v_{1:t-1}, h_{t-1}) \tag{23.2.6}$$

$$= \sum_{h_{t-1}} p(v_t|h_t)p(h_t|h_{t-1})p(h_{t-1}, v_{1:t-1}) \tag{23.2.7}$$

The cancellations follow from the conditional independence assumptions of the model. Hence if we define

$$\alpha(h_t) = p(h_t, v_{1:t}) \tag{23.2.8}$$

equation (23.2.7) above gives the *α-recursion*

$$\alpha(h_t) = \underbrace{p(v_t|h_t)}_{\text{corrector}} \underbrace{\sum_{h_{t-1}} p(h_t|h_{t-1})\alpha(h_{t-1})}_{\text{predictor}}, \qquad t > 1 \tag{23.2.9}$$

with

$$\alpha(h_1) = p(h_1, v_1) = p(v_1|h_1)p(h_1) \tag{23.2.10}$$

This recursion has the interpretation that the filtered distribution $\alpha(h_{t-1})$ is propagated forwards by the dynamics for one timestep to reveal a new 'prior' distribution at time t. This distribution is then modulated by the observation v_t, incorporating the new evidence into the filtered distribution (this is also referred to as a predictor-corrector method). Since each α is smaller than 1, and the recursion involves multiplication by terms less than 1, the α's can become very small. To avoid numerical problems it is therefore advisable to work with $\log \alpha(h_t)$, see `HMMforward.m`.

Normalisation gives the *filtered posterior*

$$p(h_t|v_{1:t}) \propto \alpha(h_t) \tag{23.2.11}$$

If we only require the filtered posterior we are free to rescale the α's as we wish. In this case an alternative to working with $\log \alpha$ messages is to work with normalised α messages so that the sum of the components is always 1.

We can write equation (23.2.7) above directly as a recursion for the filtered distribution

$$p(h_t|v_{1:t}) \propto \sum_{h_{t-1}} p(v_t|h_t)p(h_t|h_{t-1})p(h_{t-1}|v_{1:t-1}) \qquad t > 1 \tag{23.2.12}$$

Intuitively, the term $p(h_{t-1}|v_{1:t-1})$ has the effect of removing all nodes in the graph before time $t-1$ and replacing their influence by a modified 'prior' distribution on h_t. One may interpret $p(v_t|h_t)p(h_t|h_{t-1})$ as a likelihood, giving rise to the joint posterior $p(h_t, h_{t-1}|v_{1:t})$ under Bayesian updating. At the next timestep the previous posterior becomes the new prior.

23.2.3 Parallel smoothing $p(h_t|v_{1:T})$

There are two main approaches to computing $p(h_t|v_{1:T})$. Perhaps the most common in the HMM literature is the parallel method which is equivalent to message passing on factor graphs. In this one separates the smoothed posterior into contributions from the past and future:

$$p(h_t, v_{1:T}) = p(h_t, v_{1:t}, v_{t+1:T}) = \underbrace{p(h_t, v_{1:t})}_{\text{past}} \underbrace{p(v_{t+1:T}|h_t, v_{1:t})}_{\text{future}} = \alpha(h_t)\beta(h_t) \tag{23.2.13}$$

The term $\alpha(h_t)$ is obtained from the 'forward' α recursion, (23.2.9). The term $\beta(h_t)$ may be obtained using a 'backward' β recursion as we show below. The forward and backward recursions are independent and may therefore be run in parallel, with their results combined to obtain the smoothed posterior. This approach is also sometimes termed the *two-filter smoother*.

The β recursion

$$p(v_{t:T}|h_{t-1}) = \sum_{h_t} p(v_t, v_{t+1:T}, h_t|h_{t-1}) \tag{23.2.14}$$

$$= \sum_{h_t} p(v_t|\cancel{v_{t+1:T}}, h_t, \cancel{h_{t-1}})p(v_{t+1:T}, h_t|h_{t-1}) \tag{23.2.15}$$

$$= \sum_{h_t} p(v_t|h_t)p(v_{t+1:T}|h_t, \cancel{h_{t-1}})p(h_t|h_{t-1}) \tag{23.2.16}$$

Defining

$$\beta(h_t) \equiv p(v_{t+1:T}|h_t) \tag{23.2.17}$$

equation (23.2.16) above gives the *β-recursion*

$$\beta(h_{t-1}) = \sum_{h_t} p(v_t|h_t)p(h_t|h_{t-1})\beta(h_t), \qquad 2 \leq t \leq T \tag{23.2.18}$$

with $\beta(h_T) = 1$. As for the forward pass, working in log space is recommended to avoid numerical difficulties. If one only desires posterior distributions, one can also perform local normalisation at each stage since only the relative magnitude of the components of β are of importance. The smoothed posterior is then given by

$$p(h_t|v_{1:T}) \equiv \gamma(h_t) = \frac{\alpha(h_t)\beta(h_t)}{\sum_{h_t} \alpha(h_t)\beta(h_t)} \tag{23.2.19}$$

Together the $\alpha - \beta$ recursions are called the *Forward-Backward* algorithm.

23.2.4 Correction smoothing

An alternative to the parallel method is to form a recursion directly for the smoothed posterior. This can be achieved by recognising that conditioning on the present makes the future redundant:

$$p(h_t|v_{1:T}) = \sum_{h_{t+1}} p(h_t, h_{t+1}|v_{1:T}) = \sum_{h_{t+1}} p(h_t|h_{t+1}, v_{1:t}, \cancel{v_{t+1:T}})p(h_{t+1}|v_{1:T}) \tag{23.2.20}$$

This gives a recursion for $\gamma(h_t) \equiv p(h_t|v_{1:T})$:

$$\gamma(h_t) = \sum_{h_{t+1}} p(h_t|h_{t+1}, v_{1:t})\gamma(h_{t+1}) \tag{23.2.21}$$

with $\gamma(h_T) \propto \alpha(h_T)$. The term $p(h_t|h_{t+1}, v_{1:t})$ may be computed using the filtered results $p(h_t|v_{1:t})$:

$$p(h_t|h_{t+1}, v_{1:t}) \propto p(h_{t+1}, h_t|v_{1:t}) \propto p(h_{t+1}|h_t)p(h_t|v_{1:t}) \tag{23.2.22}$$

where the proportionality constant is found by normalisation. This is a form of *dynamics reversal*, as if we are reversing the direction of the hidden to hidden arrow in the HMM. This procedure, also termed the *Rauch-Tung-Striebel* smoother[1], is sequential since we need to first complete the α recursions, after which the γ recursion may begin. This is a so-called *correction smoother* since it 'corrects' the filtered result. Interestingly, once filtering has been carried out, the evidential states $v_{1:T}$ are not needed during the subsequent γ recursion.

The $\alpha - \beta$ and $\alpha - \gamma$ recursions are related through

$$\gamma(h_t) \propto \alpha(h_t)\beta(h_t) \tag{23.2.23}$$

Computing the pairwise marginal $p(h_t, h_{t-1}|v_{1:T})$

To implement the EM algorithm for learning, section(23.3.1), we require terms such as $p(h_t, h_{t-1}|v_{1:T})$. These can be obtained by message passing on either a factor graph or junction tree (for which the pairwise marginals are contained in the cliques, see fig(23.4b)). Alternatively, an explicit recursion is as follows:

$$\begin{aligned} p(h_t, h_{t+1}|v_{1:T}) &\propto p(v_{1:t}, v_{t+1}, v_{t+2:T}, h_{t+1}, h_t) \\ &= p(v_{t+2:T}|\cancel{v_{1:t}, v_{t+1}, h_t}, h_{t+1})p(v_{1:t}, v_{t+1}, h_{t+1}, h_t) \\ &= p(v_{t+2:T}|h_{t+1})p(v_{t+1}|\cancel{v_{1:t}, h_t}, h_{t+1})p(v_{1:t}, h_{t+1}, h_t) \\ &= p(v_{t+2:T}|h_{t+1})p(v_{t+1}|h_{t+1})p(h_{t+1}|\cancel{v_{1:t}}, h_t)p(v_{1:t}, h_t) \end{aligned} \tag{23.2.24}$$

Rearranging, we therefore have

$$p(h_t, h_{t+1}|v_{1:T}) \propto \alpha(h_t)p(v_{t+1}|h_{t+1})p(h_{t+1}|h_t)\beta(h_{t+1}) \tag{23.2.25}$$

See `HMMsmooth.m`.

The likelihood $p(v_{1:T})$

The likelihood of a sequence of observations can be computed from

$$p(v_{1:T}) = \sum_{h_T} p(h_T, v_{1:T}) = \sum_{h_T} \alpha(h_T) \tag{23.2.26}$$

An alternative computation can be found by making use of the decomposition

$$p(v_{1:T}) = \prod_{t=1}^{T} p(v_t|v_{1:t-1}) \tag{23.2.27}$$

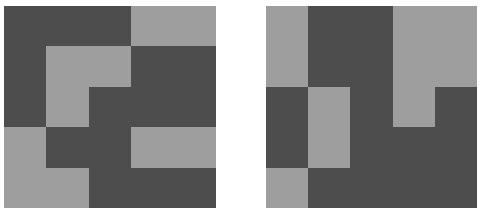
Each factor can be computed using

$$p(v_t|v_{1:t-1}) = \sum_{h_t} p(v_t, h_t|v_{1:t-1}) \tag{23.2.28}$$

$$= \sum_{h_t} p(v_t|h_t, \cancel{v_{1:t-1}})p(h_t|v_{1:t-1}) \tag{23.2.29}$$

$$= \sum_{h_t} p(v_t|h_t) \sum_{h_{t-1}} p(h_t|h_{t-1}, \cancel{v_{1:t-1}})p(h_{t-1}|v_{1:t-1}) \tag{23.2.30}$$

[1] It is most common to use this terminology for the continuous variable case, though we adopt it here also for the discrete variable case.



(a) 'Creaks' (b) 'Bumps'

Figure 23.6: Localising the burglar. The latent variable $h_t \in \{1, \ldots, 25\}$ denotes the positions, defined over the 5×5 grid of the ground floor of the house. **(a)**: A representation of the probability that the 'floor will creak' at each of the 25 positions, $p(v^{creak}|h)$. Light squares represent probability 0.9 and dark square 0.1. **(b)**: A representation of the probability $p(v^{bump}|h)$ that the burglar will bump into something in each of the 25 positions.

where the final term $p(h_{t-1}|v_{1:t-1})$ is the filtered result.

In both approaches the likelihood of an output sequence requires only a forward computation (filtering). If required, one can also compute the likelihood using, (23.2.13),

$$p(v_{1:T}) = \sum_{h_t} \alpha(h_t)\beta(h_t) \tag{23.2.31}$$

which is valid for any $1 \leq t \leq T$.

23.2.5 Most likely joint state

The most likely path $h_{1:T}$ of $p(h_{1:T}|v_{1:T})$ is the same as the most likely state (for fixed $v_{1:T}$) of

$$p(h_{1:T}, v_{1:T}) = \prod_t p(v_t|h_t)p(h_t|h_{t-1}) \tag{23.2.32}$$

The most likely path can be found using the max-product version of the factor graph or max-absorption on the junction tree. Alternatively, an explicit derivation can be obtained by considering:

$$\max_{h_T} \prod_{t=1}^{T} p(v_t|h_t)p(h_t|h_{t-1}) = \left\{ \prod_{t=1}^{T-1} p(v_t|h_t)p(h_t|h_{t-1}) \right\} \underbrace{\max_{h_T} p(v_T|h_T)p(h_T|h_{T-1})}_{\mu(h_{T-1})} \tag{23.2.33}$$

The message $\mu(h_{T-1})$ conveys information from the end of the chain to the penultimate timestep. We can continue in this manner, defining the recursion

$$\mu(h_{t-1}) = \max_{h_t} p(v_t|h_t)p(h_t|h_{t-1})\mu(h_t), \qquad 2 \leq t \leq T \tag{23.2.34}$$

with $\mu(h_T) = 1$. This means that the effect of maximising over $h_2, \ldots, h_T$ is compressed into a message $\mu(h_1)$ so that the most likely state h_1^* is given by

$$h_1^* = \operatorname*{argmax}_{h_1} p(v_1|h_1)p(h_1)\mu(h_1) \tag{23.2.35}$$

Once computed, backtracking gives

$$h_t^* = \operatorname*{argmax}_{h_t} p(v_t|h_t)p(h_t|h_{t-1}^*)\mu(h_t) \tag{23.2.36}$$

This special case of the max-product algorithm is called the *Viterbi algorithm*. Similarly, one may use the N-max-product algorithm, section(5.2.1), to obtain the N-most likely hidden paths.

Example 97 (A localisation example)**.** You're asleep upstairs in your house and awoken by noises from downstairs. You realise that a burglar is on the ground floor and attempt to understand where he his from listening to his movements. You mentally partition the ground floor into a 5×5 grid. For each grid position you know the probability that if someone is in that position the floorboard will creak, fig(23.6a).

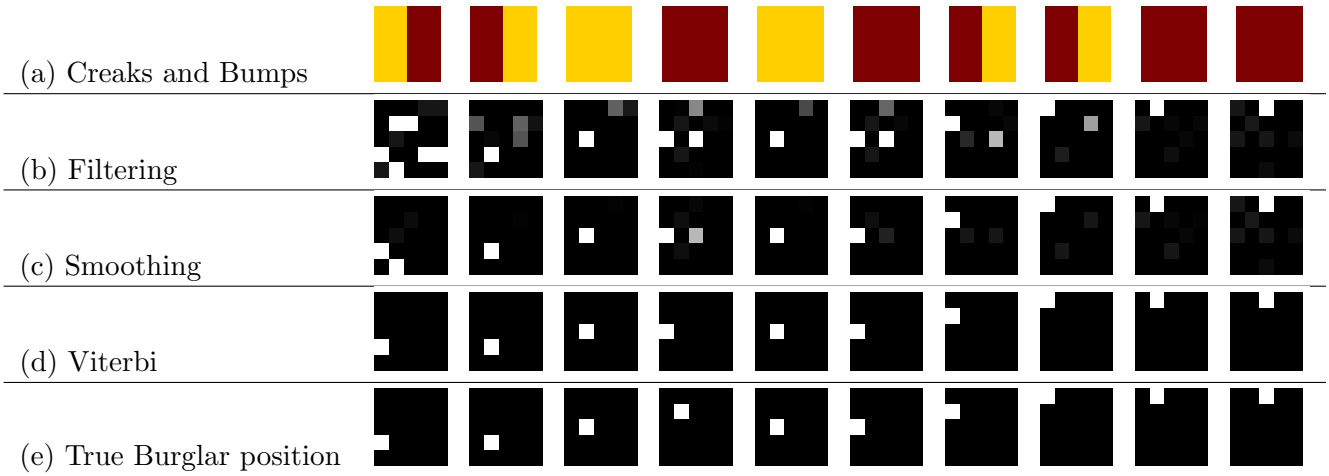


Figure 23.7: Localising the burglar through time for 10 time steps. **(a)**: Each panel represents the visible information $v_t = \left(v_t^{creak}, v_t^{bump}\right)$, where $v_t^{creak} = 1$ means that there was a 'creak in the floorboard' ($v_t^{creak} = 2$ otherwise) and $v_t^{bump} = 1$ meaning 'bumped into something' (and is in state 2 otherwise). There are 10 panels, one for each time $t = 1, \ldots, 10$. The left half of the panel represents v_t^1 and the right half v_t^2. The lighter colour represents the occurrence of a creak or bump, the darker colour the absence. **(b)**: The filtered distribution $p(h_t|v_{1:t})$ representing where we think the burglar is. **(c)**: The smoothed distribution $p(h_t|v_{1:10})$ so that we can figure out where we think the burglar went. **(d)**: The most likely (Viterbi) burglar path $\arg\max_{h_{1:10}} p(h_{1:10}|v_{1:10})$. **(e)**: The actual path of the burglar.

Similarly you know for each position the probability that someone will bump into something in the dark, fig(23.6b). The floorboard creaking and bumping into objects can occur independently. In addition you assume that the burglar will move only one grid square – forwards, backwards, left or right in a single timestep. Based on a series of bump/no bump and creak/no creak information, fig(23.7a), you try to figure out based on your knowledge of the ground floor, where the burglar might be.

We can represent the scenario using a HMM where $h \in \{1, \ldots, 25\}$ denotes the grid square. The visible variable has a factorised form $v = v^{creak} \otimes v^{bump}$ and, to use our standard code, we form a new visible variable with 4 states using

$$p(v|h) = p(v^{creak}|h)p(v^{bump}|h) \tag{23.2.37}$$

Based on the past information, our belief as to where the burglar might be is represented by the filtered distribution $p(h_t|v_{1:t})$, fig(23.7). After the burglar has left at $T = 10$, the police arrive and try to piece together where the burglar went, based on the sequence of creaks and bumps you provide. At any time t, the information as to where the burglar could have been is represented by the smoothed distribution $p(h_t|v_{1:10})$. The police's single best-guess for the sequence of burglar positions is provided by the most likely joint hidden state $\arg\max_{h_{1:10}} p(h_{1:10}|v_{1:10})$.

23.2.6 Self localisation and kidnapped robots

A robot has an internal grid-based map of its environment and for each location $h \in \{1, \ldots, H\}$, knows the likely sensor readings he would expect in that location. The robot is 'kidnapped' and placed somewhere in the environment. The robot then starts to move, gathering sensor information. Based on these readings $v_{1:t}$ and intended movements $m_{1:t}$, the robot attempts to figure out his location. Due to wheel slippage on the floor an intended action by the robot, such as 'move forwards', might not be successful. Given all the information the robot has, he would like to infer $p(h_t|v_{1:t}, m_{1:t})$. This problem differs from the burglar scenario in that the robot now has knowledge of the intended movements he makes. This should give more

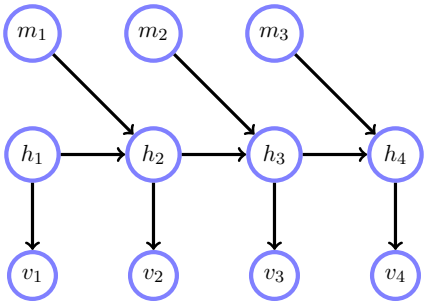


Figure 23.8: A model for robot self-localisation. At each time the robot makes an intended movement, m_t. As a generative model, knowing the intended movement m_t and the current grid position h_t, the robot has an idea of where he should be at the next time-step and what sensor reading v_{t+1} he would expect there. Based on only the sensor information $v_{1:T}$ and the intended movements $m_{1:T}$, the task is to infer a distribution over robot locations $p(h_{1:T}|m_{1:T}, v_{1:T})$.

information as to where he could be. One can view this as extra 'visible' information, though it is more natural to think of this as additional input information. A model of this scenario is, see fig(23.8),

$$p(v_{1:T}, m_{1:T}, h_{1:T}) = \prod_{t=1}^{T} p(v_t|h_t)p(h_t|h_{t-1}, m_{t-1})p(m_t) \tag{23.2.38}$$

The visible variables $v_{1:T}$ are known, as are the intended movements $m_{1:T}$. The model expresses that the movements selected by the robot are random (hence no decision making in terms of where to go next). We assume that the robot has full knowledge of the conditional distributions defining the model (he knows the 'map' of his environment and all state transition and emission probabilities). If our interest is only in localising the robot, since the inputs m are known, this model is in fact a form of time-dependent HMM:

$$p(v_{1:T}, h_{1:T}) = \prod_{t=1}^{T} p(v_t|h_t)p(h_t|h_{t-1}, t) \tag{23.2.39}$$

for a time-dependent transition $p(h_t|h_{t-1}, t)$ defined by the intended movement m_{t-1}. Any inference task required then follows the standard stationary HMM algorithms, albeit on replacing the time-independent transitions $p(h_t|h_{t-1})$ with the known time-dependent transitions.

In *self localisation and mapping* (SLAM) the robot does not know the map of his environment. This corresponds to having to learn the transition and emission distributions on-the-fly as he explores the environment.

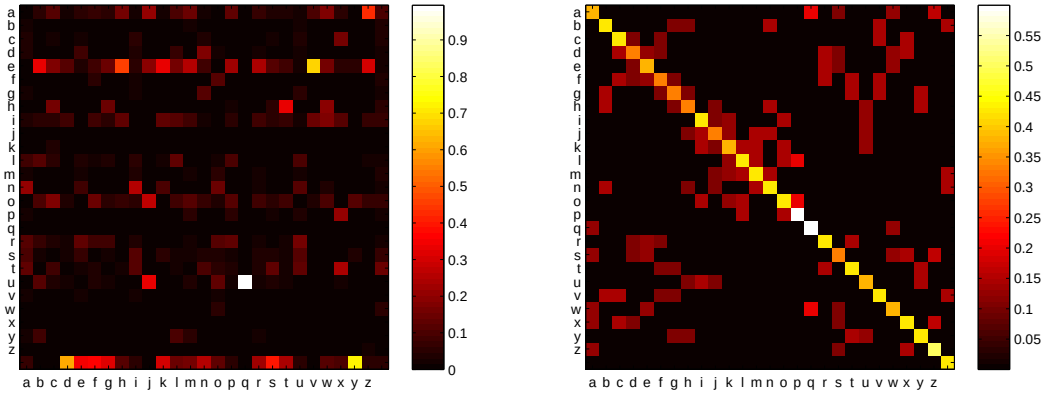
23.2.7 Natural language models

A simple generative model of language can be obtained from the letter-to-letter transitions (a so-called *bigram*). In the example below, we use this in a HMM to clean up the mis-typings.

Example 98 (Stubby fingers)**.** A 'stubby fingers' typist has the tendency to hit either the correct key or a neighbouring key. For simplicity we assume that there are 27 keys: lower case a to lower case z and the space bar. To model this we use an emission distribution $B_{ij} = p(v = i|h = j)$ where $i = 1, \ldots, 27$, $j = 1, \ldots, 27$, as depicted in fig(23.9). A database of letter-to-next-letter frequencies (`www.data-compression.com/english.shtml`), yields the transition matrix $A_{ij} = p(h' = i|h = j)$ in English. For simplicity we assume that $p(h_1)$ is uniform. Also we assume that each intended key press results in a single press. Given a typed sequence `kezrninh` what is the most likely word that this corresponds to? By listing the 200 most likely hidden sequences (using the N-max-product algorithm) and discarding those that are not in a standard English dictionary (`www.curlewcommunications.co.uk/wordlist.html`), the most likely word that was intended is `learning`. See `demoHMMbigram.m`.

23.3 Learning HMMs

Given a set of data $\mathcal{V} = \{\mathbf{v}^1, \ldots, \mathbf{v}^N\}$ of N sequences, where sequence $\mathbf{v}^n = v^n_{1:T_n}$ is of length T_n, we seek the HMM transition matrix $\mathbf{A}$, emission matrix $\mathbf{B}$, and initial vector $\mathbf{a}$ most likely to have have generated



(a) (b)

Figure 23.9: **(a)**: The letter-to-letter transition matrix for English $p(h' = i|h = j)$. **(b)**: The letter emission matrix for a typist with 'stubby fingers' in which the key or its neighbours on the keyboard are likely to be hit.

$\mathcal{V}$. We make the i.i.d. assumption so that each sequence is independently generated and assume that we know the number of hidden states H. For simplicity we concentrate here on the case of discrete visible variables, assuming also we know the number of states V.

23.3.1 EM algorithm

The application of EM to the HMM model is called the *Baum-Welch* algorithm and follows the general strategy outlined in section(11.2).

M-step

Assuming i.i.d. data, the M-step is given by maximising the 'energy':

$$\sum_{n=1}^{N} \langle \log p(v_1^n, v_2^n \ldots, v_{T^n}^n, h_1^n, h_2^n, \ldots, h_{T^n}^n) \rangle_{p^{old}(\mathbf{h}^n|\mathbf{v}^n)} \tag{23.3.1}$$

with respect to the parameters $\mathbf{A}, \mathbf{B}, \mathbf{a}$; $\mathbf{h}^n$ denotes $h_{1:T_n}$. Using the form of the HMM, we obtain

$$\sum_{n=1}^{N} \left\{ \langle \log p(h_1) \rangle_{p^{old}(h_1|\mathbf{v}^n)} + \sum_{t=1}^{T_n-1} \langle \log p(h_{t+1}|h_t) \rangle_{p^{old}(h_t,h_{t+1}|\mathbf{v}^n)} + \sum_{t=1}^{T_n} \langle \log p(v_t^n|h_t) \rangle_{p^{old}(h_t|\mathbf{v}^n)} \right\} \tag{23.3.2}$$

where for compactness we drop the sequence index from the h variables. To avoid potential confusion, we write $p^{new}(h_1 = i)$ to denote the (new) table entry for the probability that the initial hidden variable is in state i. Optimising equation (23.3.2) with respect to $p(h_1)$, (and enforcing $p(h_1)$ to be a distribution) we obtain

$$a_i^{new} \equiv p^{new}(h_1 = i) = \frac{1}{N} \sum_{n=1}^{N} p^{old}(h_1 = i|\mathbf{v}^n) \tag{23.3.3}$$

which is the average number of times that the first hidden variable is in state i. Similarly,

$$A_{i',i}^{new} \equiv p^{new}(h_{t+1} = i'|h_t = i) \propto \sum_{n=1}^{N} \sum_{t=1}^{T_n-1} p^{old}(h_t = i, h_{t+1} = i'|\mathbf{v}^n) \tag{23.3.4}$$

which is the number of times that a transition from hidden state i to hidden state i' occurs, averaged over all times (since we assumed stationarity) and training sequences. Normalising, we obtain

$$A_{i',i}^{new} = \frac{\sum_{n=1}^{N} \sum_{t=1}^{T_n-1} p^{old}(h_t = i, h_{t+1} = i'|\mathbf{v}^n)}{\sum_{i'} \sum_{n=1}^{N} \sum_{t=1}^{T_n-1} p^{old}(h_t = i, h_{t+1} = i'|\mathbf{v}^n)} \tag{23.3.5}$$

Finally,

$$B_{j,i}^{new} \equiv p^{new}(v_t = j|h_t = i) \propto \sum_{n=1}^{N}\sum_{t=1}^{T_n} \mathbb{I}\left[v_t^n = j\right] p^{old}(h_t = i|\mathbf{v}^n) \tag{23.3.6}$$

which is the expected number of times that, for the observation being in state j, the hidden state is i. The proportionality constant is determined by the normalisation requirement.

E-step

In computing the M-step above the quantities $p^{old}(h_1 = i|\mathbf{v}^n)$, $p^{old}(h_t = i, h_{t+1} = i'|\mathbf{v}^n)$ and $p^{old}(h_t = i|\mathbf{v}^n)$ are obtained by inference using the techniques described in section(23.2.1).

Equations (23.3.3,23.3.5,23.3.6) are repeated until convergence. See `HMMem.m` and `demoHMMlearn.m`.

Parameter initialisation

The EM algorithm converges to a local maxima of the likelihood and, in general, there is no guarantee that the algorithm will find the global maximum. How best to initialise the parameters is a thorny issue, with a suitable initialisation of the emission distribution often being critical for success[229]. A practical strategy is to initialise the emission $p(v|h)$ based on first fitting a simpler non-temporal mixture model $\sum_h p(v|h)p(h)$ to the data.

Continuous observations

For a continuous vector observation $\mathbf{v}_t$, with $\dim \mathbf{v}_t = D$, we require a model $p(\mathbf{v}_t|h_t)$ mapping the discrete state h_t to a distribution over outputs. Using a continuous output does not change any of the standard inference message passing equations so that inference can be carried out for essentially arbitrarily complex emission distributions. Indeed, filtering, smoothing and Viterbi inference, the normalisation Z of the emission $p(v|h) = \phi(v, h)/Z$ is not required. For learning, however, the emission normalisation constant is required since this is a dependent on the parameters of the model.

23.3.2 Mixture emission

To make a richer emission model (particularly for continuous observations), one approach is use a mixture

$$p(v_t|h_t) = \sum_{k_t} p(v_t|k_t, h_t)p(k_t|h_t) \tag{23.3.7}$$

where k_t is a discrete summation variable. For learning, it is useful to consider the k_t as additional latent variables so that updates for each component of the emission model can be derived. To achieve this, consider the contribution to the energy from the emission (assuming equal length sequences):

$$E_v \equiv \sum_n \sum_{t=1}^{T} \left\langle \log p(v_t^n|h_t)\right\rangle_{q(h_t|v_{1:T}^n)} \tag{23.3.8}$$

As it stands, the parameters of each component $p(v_t|k_t, h_t)$ are coupled in the above expression. One approach is to consider

$$\mathrm{KL}(q(k_t|h_t)|p(k_t|h_t, v_t)) \geq 0 \tag{23.3.9}$$

from which we immediately obtain the bound

$$\log p(v_t, h_t) \geq -\left\langle \log q(k_t|h_t)\right\rangle_{q(k_t|h_t)} + \left\langle \log p(v_t, k_t, h_t)\right\rangle_{q(k_t|h_t)} \tag{23.3.10}$$

and

$$\log p(v_t^n|h_t^n) \geq -\left\langle \log q(k_t|h_t^n)\right\rangle_{q(k_t|h_t^n)} + \left\langle \log p(v_t^n|k_t, h_t^n)\right\rangle_{q(k_t|h_t^n)} + \left\langle \log p(k_t|h_t^n)\right\rangle_{q(k_t|h_t^n)} \tag{23.3.11}$$

Using this in the energy contribution (23.3.8) we have the bound on the energy contribution

$$E_v \geq \sum_n \sum_{t=1}^{T} \Big\langle -\langle \log q(k_t|h_t^n)\rangle_{q(k_t|h_t^n)} + \langle \log p(v_t^n|k_t,h_t^n)\rangle_{q(k_t|h_t^n)} + \langle \log p(k_t|h_t^n)\rangle_{q(k_t|h_t^n)} \Big\rangle_{q(h_t^n|v_{1:T}^n)} \quad (23.3.12)$$

We may now maximise this lower bound on the energy (instead of the energy itself). The contribution from each emission component $p(v=\mathsf{v}|h=\mathsf{h},k=\mathsf{k})$ is

$$\sum_n \sum_{t=1}^{T} q(k_t=\mathsf{k}|h_t^n=\mathsf{h}) q(h_t^n=\mathsf{h}|v_{1:T}^n) \log p(v_t^n|h=\mathsf{h},k=\mathsf{k}) \quad (23.3.13)$$

The above can then be optimised (M-step) for fixed $q(k_t=\mathsf{k}|h_t^n=\mathsf{h})$, with these distributions updated using

$$q^{new}(k_t|h_t^n) \propto p(v^n|h_t^n,k_t)p(k_t|h_t) \quad (23.3.14)$$

The contribution to the energy bound from the mixture weights is given by

$$\log p(k=\mathsf{k}|h=\mathsf{h}) \sum_n \sum_{t=1}^{T} q(k_t=\mathsf{k}|h_t^n=\mathsf{h}) q(h_t^n=\mathsf{h}|v_{1:T}^n) \quad (23.3.15)$$

so that the M-step update for the mixture weights is,

$$p(k=\mathsf{k}|h=\mathsf{h}) \propto \sum_n \sum_{t=1}^{T} q(k_t=\mathsf{k}|h_t^n=\mathsf{h}) q(h_t^n=\mathsf{h}|v_{1:T}^n) \quad (23.3.16)$$

In this case the EM algorithm is composed of an 'emission' EM loop in which the transitions and $q(h_t^n=\mathsf{h}|v_{1:T}^n)$ is fixed, during which the emissions $p(v|h,k)$ are learned, along with updating $q(k_t=\mathsf{k}|h_t^n=\mathsf{h})$. The 'transition' EM loop fixes the emission distribution $p(v|h)$ and learns the best transition $p(h_t|h_{t-1})$.

An alternative to the above derivation is to consider the k as hidden variables, and then use standard EM algorithm on the joint latent variables (h_t,k_t). The reader may show that the two approaches are equivalent.

23.3.3 The HMM-GMM

A common continuous observation mixture emission model component is a Gaussian

$$p(\mathbf{v}_t|k_t,h_t) = \mathcal{N}\left(\mathbf{v}_t|\boldsymbol{\mu}_{k_t,h_t},\boldsymbol{\Sigma}_{k_t,h_t}\right) \quad (23.3.17)$$

so that k_t,h_t indexes the $K\times H$ mean vectors and covariance matrices. EM updates for these means and covariances are straightforward to derive from equation (23.3.12), see exercise(232). These models are common in tracking applications, in particular in speech recognition (usually under the constraint that the covariances are diagonal).

23.3.4 Discriminative training

HMMs can be used for supervised learning of sequences. That is, for each sequence $v_{1:T}^n$, we have a corresponding class label c^n. For example, we might associated a particular composer c with a sequence $v_{1:T}$ and wish to make a model that will predict the composer for a novel music sequence. A generative approach to using HMMs for classification is to train a separate HMM for each class, $p(v_{1:T}|c)$ and subsequently use Bayes' rule to form the classification for a novel sequence $v_{1:T}^*$ using

$$p(c^*|v_{1:T}^*) = \frac{p(v_{1:T}^*|c^*)p(c^*)}{\sum_{c'=1}^{C} p(v_{1:T}^*|c')p(c')} \quad (23.3.18)$$

If the data is noisy and difficult to model, however, this generative approach may not work well since much of the expressive power of each model is used to model the complex data, rather than focussing on

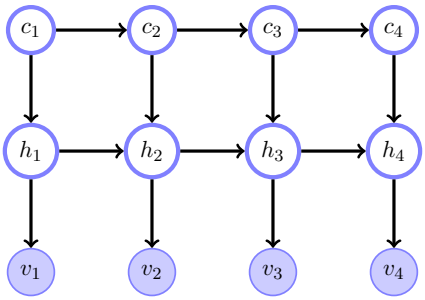


Figure 23.10: An explicit duration HMM. The counter variables c_t deterministically count down to zero. When they reach one, a h transition is allowed, and the new value for c_t is sampled.

the decision boundary. In applications such as speech recognition, improvements in performance are often reported when the models are trained in a discriminative way. In discriminative training, see for example [150], one defines a new single discriminative model, formed from the C HMMs using

$$p(c|v_{1:T}) = \frac{p(v_{1:T}|c)p(c)}{\sum_{c'=1}^{C} p(v_{1:T}|c')p(c')} \tag{23.3.19}$$

and then maximises the likelihood of a set of observed classes and corresponding observations $v_{1:T}$. For a single data pair, $(c^n, v^n_{1:T})$, the log likelihood is

$$\log p(c^n|v^n_{1:T}) = \underbrace{\log p(v^n_{1:T}|c)}_{\text{generative likelihood}} + \log p(c) - \log \sum_{c'=1}^{C} p(v^n_{1:T}|c')p(c') \tag{23.3.20}$$

The first term above represents the generative likelihood term, with the last term accounting for the discrimination. Whilst deriving EM style updates is hampered by the discriminative terms, computing the gradient is straightforward using the technique described in section(11.7).

In some applications, a class label c_t is available at each timestep, together with an observation v_t. Given a training sequence $v_{1:T}, c_{1:T}$ (or more generally a set of sequences) the aim is to find the optimal class sequence $c^*_{1:T}$ for a novel observation sequence $v^*_{1:T}$. One approach is to train a generative model

$$p(v_{1:T}, c_{1:T}) = \prod_t p(v_t|c_t)p(c_t|c_{t-1}) \tag{23.3.21}$$

and subsequently use Viterbi to form the class $c^*_{1:T} = \arg\max_{c_{1:T}} p(c_{1:T}|v_{1:T})$. However, this approach may not be optimal in terms of class discrimination. A cheap surrogate is to train a discriminative classification model $\tilde{p}(c_t|v_t)$ separately. With this one can form the emission (here written for continuous v_t)

$$p(v_t|c_t) = \frac{\tilde{p}(c_t|v_t)\tilde{p}(v_t)}{\int_{v_t} \tilde{p}(c_t|v_t)\tilde{p}(v_t)} \tag{23.3.22}$$

where $\tilde{p}(v_t)$ is user defined. Whilst computing the local normalisation $\int_{v_t} \tilde{p}(c_t|v_t)\tilde{p}(v_t)$ may be problematic, if the only use of $p(v_t|c_t)$ is to find the optimal class sequence for a novel observation sequence $v^*_{1:T}$,

$$c^*_{1:T} = \underset{c_{1:T}}{\text{argmax}}\ p(c_{1:T}|v^*_{1:T}) \tag{23.3.23}$$

then the local normalisations play no role since they are independent of c. Hence, during Viterbi decoding we may replace the term $p(v_t|h_t)$ with $\tilde{p}(c_t|v_t)$ without affecting the optimal sequence. Using a model in this way is a special case of the general hybrid procedure described in section(13.2.4). The approach is suboptimal since learning the classifier is divorced from learning the transition model. Nevertheless, this heuristic historically has some support in the speech recognition community.

23.4 Related Models

23.4.1 Explicit duration model

For a HMM with self-transition $p(h_t = i|h_{t-1} = i) \equiv \gamma_i$, the probability that the latent dynamics stays in state i for τ timesteps is γ_i^τ, which decays exponentially with time. In practice, however, we would

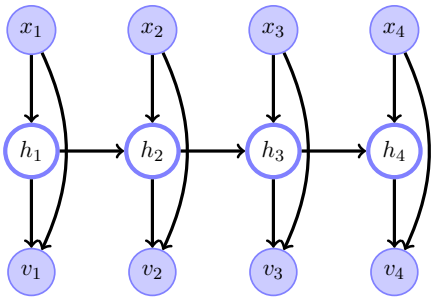


Figure 23.11: A first order input-output hidden Markov model. The input x and output v nodes are shaded to emphasise that their states are known during training. During testing, the inputs are known and the outputs are predicted.

often like to constrain the dynamics to remain in the same state for a minimum number of timesteps, or to have a specified duration distribution. A way to enforce this is to use a latent counter variable c_t which at the beginning is initialised to a duration sampled from the duration distribution $p_{dur}(c_t)$ with maximal duration D_{max}. Then at each timestep the counter decrements by 1, until it reaches 1, after which a new duration is sampled:

$$p(c_t|c_{t-1}) = \begin{cases} \delta(c_t, c_{t-1} - 1) & c_{t-1} > 1 \\ p_{dur}(c_t) & c_{t-1} = 1 \end{cases} \tag{23.4.1}$$

The state h_t can transition only when $c_t = 1$:

$$p(h_t|h_{t-1}, c_t) = \begin{cases} \delta(h_t, h_{t-1}) & c_t > 1 \\ p_{tran}(h_t|h_{t-1}) & c_t = 1 \end{cases} \tag{23.4.2}$$

Including the counter variable c defines a joint latent variable $(c_{1:T}, h_{1:T})$ distribution that ensures h remains in a desired minimal number of timesteps, see fig(23.10). Since $\dim c_t \otimes h_t = D_{max}H$, naively the computational complexity of inference in this model scales as $O\left(TH^2D_{max}^2\right)$. However, when one runs the forward and backward recursions, the deterministic nature of the transitions means that this can be reduced to $O\left(TH^2D_{max}\right)$[199] – see also exercise(233).

The hidden semi-Markov model generalises the explicit duration model in that once a new duration c_t is sampled, the model emits a distribution $p(v_{t:t+c_t-1}|h_t)$ defined on a segment of the next c_t observations[216].

23.4.2 Input-Output HMM

The IOHMM[31] is a HMM with additional input variables $x_{1:T}$, see fig(23.11). Each input can be continuous or discrete and modulates the transitions

$$p(v_{1:T}, h_{1:T}|x_{1:T}) = \prod_t p(v_t|h_t, x_t)p(h_t|h_{t-1}, x_t) \tag{23.4.3}$$

The IOHMM may be used as a conditional predictor, where the outputs v_t represent the prediction at time t. In the case of continuous inputs and discrete outputs, the tables $p(v_t|h_t, x_t)$ and $p(h_t|h_{t-1}, x_t)$ are usually parameterised using a non-linear function, for example

$$p(v_t = y|h_t = h, x_t = x, \mathbf{w}) \propto e^{\mathbf{w}_{h,y}^{\mathsf{T}} x} \tag{23.4.4}$$

Inference then follows in a similar manner as for the standard HMM. Defining

$$\alpha(h_t) \equiv p(h_t, v_{1:t}|x_{1:t}) \tag{23.4.5}$$

the forward pass is given by

$$\alpha(h_t) = \sum_{h_{t-1}} p(h_t, h_{t-1}, v_{1:t-1}, v_t|x_{1:t}) \tag{23.4.6}$$

$$= \sum_{h_{t-1}} p(v_t|v_{1:t-1}, x_{1:t}, h_t, h_{t-1})p(h_t|v_{1:t-1}, x_{1:t}, h_{t-1})p(v_{1:t-1}, h_{t-1}|x_{1:t}) \tag{23.4.7}$$

$$= p(v_t|x_t, h_t) \sum_{h_{t-1}} p(h_t|h_{t-1}, x_t)\alpha(h_{t-1}) \tag{23.4.8}$$

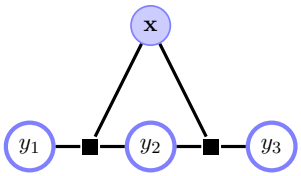


Figure 23.12: Linear chain CRF. Since the input x is observed, the distribution is just a linear chain factor graph. The inference of pairwise marginals $p(y_t, y_{t-1}|x)$ is therefore straightforward using message passing.

The γ backward pass is

$$p(h_t|x_{1:T}, v_{1:T}) = \sum_{h_{t+1}} p(h_t, h_{t+1}|x_{1:t+1}, x_{t+2:T}, v_{1:T}) = \sum_{h_{t+1}} p(h_t|h_{t+1}, x_{1:t+1}, v_{1:t}) p(h_{t+1}|x_{1:T}, v_{1:T}) \tag{23.4.9}$$

for which we need

$$p(h_t|h_{t+1}, x_{1:t+1}, v_{1:t}) = \frac{p(h_{t+1}, h_t|x_{1:t+1}, v_{1:t})}{p(h_{t+1}|x_{1:t+1}, v_{1:t})} = \frac{p(h_{t+1}|h_t, x_{t+1}) p(h_t|x_{1:t}, v_{1:t})}{\sum_{h_t} p(h_{t+1}|h_t, x_{t+1}) p(h_t|x_{1:t}, v_{1:t})} \tag{23.4.10}$$

The likelihood can be found from $\sum_{h_T} \alpha(h_T)$.

Direction bias

Consider predicting the output distribution $p(v_t|x_{1:T})$ given both past and future input information $x_{1:T}$. Because the hidden states are unobserved we have $p(v_t|x_{1:T}) = p(v_t|x_{1:t})$. Thus the prediction uses only past information and discards any future contextual information. This 'direction bias' is sometimes considered problematic (particularly in natural language modelling) and motivates the use of undirected models, such as conditional random fields.

23.4.3 Linear chain CRFs

Linear chain Conditional Random Fields (CRFs) are an extension of the unstructured CRFs we briefly discussed in section(9.4.6) and have application to modelling the distribution of a set of outputs $y_{1:T}$ given an input vector $\mathbf{x}$. For example, $\mathbf{x}$ might represent a sentence in English, and $y_{1:T}$ should represent the translation into French. Note that the vector $\mathbf{x}$ does not have to have dimension T. A first order linear chain CRF has the form

$$p(y_{1:T}|\mathbf{x}, \boldsymbol{\lambda}) = \frac{1}{Z(\mathbf{x}, \boldsymbol{\lambda})} \prod_{t=2}^{T} \phi_t(y_t, y_{t-1}, \mathbf{x}, \boldsymbol{\lambda}) \tag{23.4.11}$$

where $\boldsymbol{\lambda}$ are the free parameters of the potentials. In practice it is common to use potentials of the form

$$\exp\left(\sum_{k=1}^{K} \lambda_k f_{k,t}(y_t, y_{t-1}, \mathbf{x})\right) \tag{23.4.12}$$

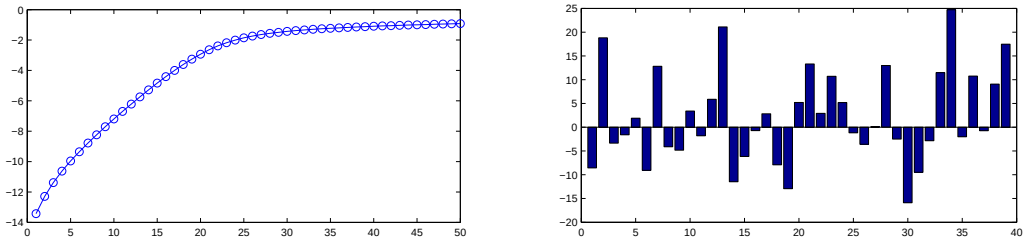
where $f_{k,t}(y_t, y_{t-1}, x)$ are 'features', see also section(9.4.6). Given a set of input-output sequence pairs, $\mathbf{x}^n, y^n_{1:T}, n = 1, \ldots, N$ (assuming all sequenced have equal length T for simplicity), we can learn the parameters $\boldsymbol{\lambda}$ by Maximum Likelihood. Under the standard i.i.d. data assumption, the log likelihood is

$$L(\boldsymbol{\lambda}) = \sum_{t,n} \sum_{k} \lambda_k f_k(y^n_t, y^n_{t-1}, \mathbf{x}^n) - \sum_{n} \log Z(\mathbf{x}^n, \boldsymbol{\lambda}) \tag{23.4.13}$$

The reader may readily check that the log likelihood is concave so that the objective function has no local optima. The gradient is given by

$$\frac{\partial}{\partial \lambda_i} L(\boldsymbol{\lambda}) = \sum_{n,t} \left(f_i(y^n_t, y^n_{t-1}, \mathbf{x}^n) - \langle f_i(y_t, y_{t-1}, \mathbf{x}^n) \rangle_{p(y_t, y_{t-1}|\mathbf{x}^n, \lambda)} \right) \tag{23.4.14}$$

Learning therefore requires inference of the marginal terms $p(y_t, y_{t-1}|\mathbf{x}, \boldsymbol{\lambda})$. Since equation (23.4.11) corresponds to a linear chain factor graph, see fig(23.12), inference of pairwise marginals is straightforward



(a) (b)

Figure 23.13: Using a linear chain CRF to learn the sequences in table(23.1). **(a)**: The evolution of the log likelihood under gradient ascent. **(b)**: The learned parameter vector $\boldsymbol{\lambda}$ at the end of training.

using message passing. This can be achieved using either the standard factor graph message passing or by deriving an explicit algorithm, see exercise(227).

Finding the most likely output sequence for a novel input $\mathbf{x}^*$ is straightforward since

$$y^*_{1:T} = \underset{y_{1:T}}{\text{argmax}} \prod_t \phi_t(y_t, y_{t-1}, \mathbf{x}^*, \boldsymbol{\lambda}) \tag{23.4.15}$$

corresponds again to a simple linear chain, for which max-product inference yields the required result, see also exercise(226).

In some applications, particularly in natural language processing, the dimension K of the vector of features $f_1, \ldots, f_K$ may be many hundreds of thousands. This means that the storage of the Hessian is not feasible for Newton based training and either limited memory methods or conjugate gradient techniques are typically preferred[288].

7	4	7	2	3	4	5	7	3	5		
3	1	3	1	2	3	3	1	2	1		
10	3	2									
1	1	1									
9	8	3	9								
2	3	3	3								
7	8	8	4	6	10	2	7	7	6	6	10
2	3	3	1	3	1	1	3	2	2	3	1
7	9	3	3	4	8	8					
3	1	2	3	3	3	3					

Table 23.1: A subset of the 10 training input-output sequences. Each row contains an input x_t (upper entry) and output y_t (lower entry). There are 10 input states and 3 output states.

Example 99 (Linear Chain CRF)**.** As a model for the data in table(23.1), a linear CRF model has potentials $\phi(y_t, y_{t-1}, x_t) = \exp\left(\sum_i \lambda_i f_i(y_t, y_{t-1}, x_t)\right)$ where we set the binary feature functions by first mapping each of the $\dim(x) \times \dim(y)^2$ states to a unique integer $i(a, b, c)$ from 1 to $\dim(x) \times \dim(y)^2$

$$f_{i(a,b,c)}(y_t, y_{t-1}, x_t) = \mathbb{I}[y_t = a]\,\mathbb{I}[y_{t-1} = b]\,\mathbb{I}[x_t = c] \tag{23.4.16}$$

That is, each joint configuration of y_t, y_{t-1}, x_t is mapped to an index, and in this case the feature vector $\mathbf{f}$ will trivially have only a single non-zero entry. The evolution of the gradient ascent training algorithm is plotted in fig(23.13). In practice one would use richer feature functions defined to seek features of the input sequence $\mathbf{x}$ and also to produce a feature vector with more than one non-zero entry. See `demoLinearCRF.m`.

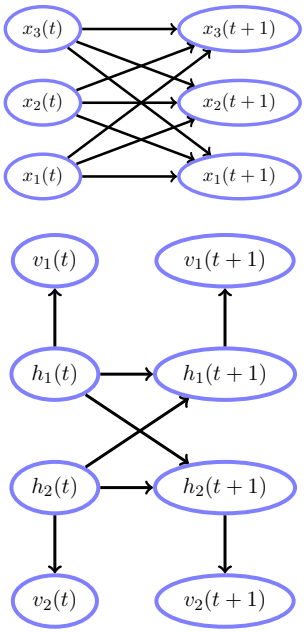


Figure 23.14: A Dynamic Bayesian Network. Possible transitions between variables at the same time-slice have not been shown.

Figure 23.15: A Coupled HMM. For example the upper HMM might model speech, and the lower the corresponding video sequence. The upper hidden units then correspond to phonemes, and the lower to mouth positions; this model therefore captures the expected coupling between mouth positions and phonemes.

23.4.4 Dynamic Bayesian networks

A DBN is defined as a belief networkreplicated through time. For a multivariate $\mathbf{x}_t$, with $\dim \mathbf{x}_t = D$, the DBN defines a joint model

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_T) = \prod_{t=1}^{T} \prod_{i=1}^{D} p(x_i(t) | \mathbf{x}_{\setminus i}(t), \mathbf{x}(t-1)) \tag{23.4.17}$$

where $\mathbf{x}_{\setminus i}(t)$ denotes the set of variables at time t, except for $x_i(t)$. The form of each $p(x_i(t)|\mathbf{x}_{\setminus i}(t), \mathbf{x}(t-1))$ is chosen such that the overall distribution remains acyclic. At each time-step t there is a set of variables $x_i(t), i = 1, \ldots, X$, some of which may be observed. In a first order DBN, each variable $x_i(t)$ has parental variables taken from the set of variables in the previous time-slice, $\mathbf{x}_{t-1}$, or from the present time-slice. In most applications, the model is temporally homogeneous so that one may fully describe the distribution in terms of a two-time-slice model, fig(23.14). The generalisation to higher-order models is straightforward. A *coupled HMM* is a special DBN that may be used to model coupled 'streams' of information, for example video and audio, see fig(23.15)[209].

23.5 Applications

23.5.1 Object tracking

HMMs are used to track moving objects, based on an understanding of the dynamics of the object (encoded in the transition distribution) and an understanding of how an object with a known position would be observed (encoded in the emission distribution). Given an observed sequence, the hidden position can then be inferred. The burglar, example(10) is a case in point. HMMs have been applied in a many tracking contexts, including tracking people in videos, musical pitch, and many more[54, 229, 51].

23.5.2 Automatic speech recognition

Many speech recognition systems make use of HMMs[300]. Roughly speaking, a continuous output vector $\mathbf{v}_t$ at time t, represents which frequencies are present in the speech signal in a small window around time t. These acoustic vectors are typically formed from taking a discrete Fourier transform of the speech signal over a small window around time t, with additional transformations to mimic human auditory processing. Alternatively, related forms of linear coding of the observed acoustic waveform may be used[131].

The corresponding discrete latent state h_t represents a phoneme – a basic unit of human speech (for which there are 44 in standard English). Training data is painstakingly constructed by a human linguist who

determines the phoneme h_t for each time t and many different observed sequences $\mathbf{v}_t$. Given then each acoustic vector $\mathbf{v}_t$ and an associated phoneme h_t, one may use maximum likelihood to fit a mixture of (usually isotropic) Gaussians $p(\mathbf{v}_t|h_t)$ to $\mathbf{v}_t$. This forms the emission distribution for a HMM.

Using the database of labelled phonemes, the phoneme transition $p(h_t|h_{t-1})$ can be learned (by simple counting) and forms the transition distribution for a HMM. Note that in this case, since the 'hidden' variable h and observation v are known during training, training the HMM is straightforward and boils down to training the emission and transition distributions independently.

For a new sequence of 'acoustic' vectors $\mathbf{v}_{1:T}$ we can then use the HMM to infer the most likely phoneme sequence through time, $\arg\max_{h_{1:T}} p(h_{1:T}|\mathbf{v}_{1:T})$, which takes into account both the way that phonemes appear as acoustic vectors, and also the prior language constraints of likely phoneme to phoneme transitions. The fact that people speak at different speeds can be addressed using *time-warping* in which the latent phoneme remains in the same state for a number of timesteps.

HMM models are typically trained on the assumption of 'clean' underlying speech. In practice noise corrupts the speech signal in a complex way, so that the resulting model is inappropriate, and performance degrades significantly. To account for this, it is traditional to attempt to denoise the signal before sending this to a standard HMM recogniser.

If the HMM is used to model a single word, it is natural to constrain the hidden state sequence to go 'forwards' through time, visiting a set of states in sequence (since the phoneme order for the word is known). In this case the structure of the transition matrices is upper triangular (or lower, depending on your definition), or even a banded triangular matrix. Such forward constraints describe a so-called *left-to-right transition matrix*.

23.5.3 Bioinformatics

In the field of Bioinformatics HMMs have been widely applied to modelling genetic sequences. Multiple sequence alignment using forms of constrained HMMs have been particularly successful. Other applications involve gene finding and protein family modelling[163, 84].

23.5.4 Part-of-speech tagging

Consider the sentence below in which each word has been linguistically tagged

```
hospitality_NN is_BEZ an_AT excellent_JJ virtue_NN ,_,
but_CC not_XNOT when_WRB the_ATI guests_NNS have_HV
to_TO sleep_VB in_IN rows_NNS in_IN the_ATI cellar_NN !_!
```

The subscripts denote a linguistic tag, for example NN is the singular common noun tag, ATI is the article tag *etc.* Given a training set of such tagged sequences, the task is to tag a novel word sequence. One approach is to use h_t to be a tag, and v_t to be a word and fit a HMM to this data. For the training data, both the tags and words are observed so that Maximum Likelihood training of the transition and emission distribution can be achieved by simple counting. Given a new sequence of words, the most likely tag sequence can be inferred using the Viterbi algorithm.

More recent part-of-speech taggers tend to use conditional random fields in which the input sequence $x_{1:T}$ is the sentence and the output sequence $y_{1:T}$ is the tag sequence. One possible parameterisation of for a linear chain CRF is to use a potential of the form $\phi(y_{t-1}, y_t)\phi(y_t, \mathbf{x})$ in which the first factor encodes the grammatical structure of the language and the second the a priori likely tag y_t[166].

23.6 Code

`demoMixMarkov.m`: Demo for Mixture of Markov models
`mixMarkov.m`: Mixture of Markov models
`demoHMMinference.m`: Demo of HMM Inference
`HMMforward.m`: Forward α recursion
`HMMbackward.m`: Forward β recursion
`HMMgamma.m`: RTS γ 'correction' recursion
`HMMsmooth.m`: Single and Pairwise $\alpha - \beta$ smoothing
`HMMviterbi.m`: Most Likely State (Viterbi) algorithm
`demoHMMburglar.m`: Demo of Burglar Localisation
`demoHMMbigram.m`: demo of stubby fingers typing
`HMMem.m`: EM algorithm for HMM (Baum-Welch)
`demoHMMlearn.m`: demo of EM algorithm for HMM (Baum-Welch)
`demoLinearCRF.m`: demo of learning a linear chain CRF

The following linear chain CRF potential is particularly simple and in practice one would use a more complex one.
`linearCRFpotential.m`: Linear CRF potential
The following likelihood and gradient routines are valid for any linear CRF potential $\phi(y_{t-1}, y_t, \mathbf{x})$.
`linearCRFgrad.m`: Linear CRF gradient
`linearCRFloglik.m`: Linear CRF log likelihood

23.7 Exercises

Exercise 217. *A stochastic matrix M_{ij} as non-negative entries with $\sum_i M_{ij} = 1$. Consider an eigenvalue λ and eigenvector $\mathbf{e}$ such $\sum_j M_{ij} e_j = \lambda e_i$. By summing over i show that, provided $\sum_i e_i > 0$, then λ must be equal to 1.*

Exercise 218. *Consider the Markov chain with transition matrix*

$$\mathbf{M} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{23.7.1}$$

Show that this Markov chain does not have an equilibrium distribution and state a stationary distribution for this chain.

Exercise 219. *Consider a HMM with 3 states ($M = 3$) and 2 output symbols, with a left-to-right state transition matrix*

$$\mathbf{A} = \begin{pmatrix} 0.5 & 0.0 & 0.0 \\ 0.3 & 0.6 & 0.0 \\ 0.2 & 0.4 & 1.0 \end{pmatrix} \tag{23.7.2}$$

where $A_{ij} \equiv p(h(t+1) = i | h(t) = j)$, emission matrix $B_{ij} \equiv p(v(t) = i | h(t) = j)$

$$\mathbf{B} = \begin{pmatrix} 0.7 & 0.4 & 0.8 \\ 0.3 & 0.6 & 0.2 \end{pmatrix} \tag{23.7.3}$$

and initial state probability vector $\mathbf{a} = (0.9\ 0.1\ 0.0)^\mathsf{T}$. Given the observed symbol sequence is $v_{1:3} = (0, 1, 1)$:

1. *Compute $p(v_{1:3})$.*
2. *Compute $p(h_1 | v_{1:3})$.*
3. *Find the most probable hidden state sequence $\arg\max_{h_{1:3}} p(h_{1:3} | v_{1:3})$.*

Exercise 220. *This exercise follows from example(98). Given the 27 long character string* `rgenmonleunosbpnntje vrancg` *typed with 'stubby fingers', what is the most likely correct English sentence intended? In the list of decoded sequences, what value is* $\log p(h_{1:27}|v_{1:27})$ *for this sequence? You will need to modify* `demoHMMbigram.m` *suitably.*

Exercise 221. *Show that if a transition probability* $A_{ij} = p(h_t = i|h_{t-1} = j)$ *in a HMM is initialised to zero for EM training, then it will remain at zero throughout training.*

Exercise 222. *Consider the problem : Find the most likely joint output sequence* $v_{1:T}$ *for a HMM. That is,*

$$v^*_{1:T} \equiv \underset{v_{1:T}}{\text{argmax}}\ p(v_{1:T}) \tag{23.7.4}$$

where

$$p(h_{1:T}, v_{1:T}) = \prod_{t=1}^{T} p(v_t|h_t)p(h_t|h_{t-1}) \tag{23.7.5}$$

1. *Explain why a local message passing algorithm cannot, in general, be found for this problem and discuss the computational complexity of finding an exact solution.*

2. *Explain how to adapt the Expectation-Maximisation algorithm to form a recursive algorithm, for finding an approximate* $v^*_{1:T}$*. Explain which it guarantees an improved solution at each iteration. Additionally, explain how the algorithm can be implemented using local message passing.*

Exercise 223. *Explain how to train a HMM using EM, but with a constrained transition matrix. In particular, explain how to learn a transition matrix with an upper triangular structure.*

Exercise 224. *Write a program to fit a mixture of* L^{th} *order Markov models.*

Exercise 225.

1. *Using the correspondence* $A = 1, C = 2, G = 3, T = 4$ *define a* 4×4 *transition matrix* p *that produces sequences of the form*

$$A, C, G, T, A, C, G, T, A, C, G, T, A, C, G, T, \ldots \tag{23.7.6}$$

Now define a new transition matrix

```
pnew = 0.9*p + 0.1*ones(4)/4
```
(23.7.7)

Define a 4×4 *transition matrix* q *that produces sequences of the form*

$$T, G, C, A, T, G, C, A, T, G, C, A, T, G, C, A, \ldots \tag{23.7.8}$$

Now define a new transition matrix

```
qnew = 0.9*q + 0.1*ones(4)/4
```
(23.7.9)

Assume that the probability of being in the initial state of the Markov chain $p(h_1)$ *is constant for all four states* A, C, G, T*. What is the probability that the Markov chain* `pnew` *generated the sequence* S *given by*

$$S \equiv A, A, G, T, A, C, T, T, A, C, C, T, A, C, G, C \tag{23.7.10}$$

2. *Similarly what is the probability that* S *was generated by* `qnew`*? Does it make sense that* S *has a higher likelihood under pnew compared with qnew?*

3. *Using the function* `randgen.m`*, generate 100 sequences of length 16 from the Markov chain defined by* `pnew`*. Similarly, generate 100 sequences each of length 16 from the Markov chain defined by* `qnew`*. Concatenate all these sequences into a cell array* `v` *so that* `v{1}` *contains the first sequence and* `v{200}` *the last sequence. Use* `MixMarkov.m` *to learn the optimum Maximum Likelihood parameters that generated these sequences. Assume that there are* $H = 2$ *kinds of Markov chain. The result returned in* `phgvn` *indicate the posterior probability of sequence assignment. Do you agree with the solution found?*

4. *Take the sequence* S *as defined in equation (23.7.10). Define an emission distribution that has 4 output states such that*

$$p(v = i|h = j) = \begin{cases} 0.7 & i = j \\ 0.1 & i \neq j \end{cases} \tag{23.7.11}$$

Using this emission distribution and the transition given by `pnew` *defined in equation (23.7.7), adapt* `demoHMMinferenceSimple.m` *suitably to find the most likely hidden sequence* $h^p_{1:16}$ *that generated the observed sequence* S*. Repeat this computation but for the transition* `qnew` *to give* $h^q_{1:16}$*. Which hidden sequence –* $h^p_{1:16}$ *or* $h^q_{1:16}$ *is to be preferred? Justify your answer.*

Exercise 226. *Derive an algorithm that will find the most likely joint state*

$$\underset{h_{1:T}}{\text{argmax}} \prod_{t=2}^{T} \phi_t(h_{t-1}, h_t) \tag{23.7.12}$$

for arbitrarily defined potentials $\phi_t(h_{t-1}, h_t)$*.*

1. *First consider*

$$\max_{h_{1:T}} \prod_{t=2}^{T} \phi_t(h_{t-1}, h_t) \tag{23.7.13}$$

Show that how the maximisation over h_T *may be pushed inside the product and that the result of the maximisation can be interpreted as a message*

$$\gamma_{T-1 \leftarrow T}(h_{T-1}) \tag{23.7.14}$$

2. *Derive the recursion*

$$\gamma_{t-1 \leftarrow t}(h_{t-1}) = \max_{h_t} \phi_t(h_t, h_{t-1}) \gamma_{t \leftarrow t+1}(h_t) \tag{23.7.15}$$

3. *Explain how the above recursion enables the computation of*

$$\underset{h_1}{\text{argmax}} \prod_{t=2}^{T} \phi_t(h_t, h_{t-1}) \tag{23.7.16}$$

4. *Explain how once the most likely state for* h_1 *is computed, one may efficiently compute the remaining optimal states* $h_2, \ldots, h_T$*.*

Exercise 227. *Derive an algorithm that will compute pairwise marginals*

$$p(h_t, h_{t-1}) \tag{23.7.17}$$

from the joint distribution

$$p(h_{1:T}) \propto \prod_{t=2}^{T} \phi_t(h_{t-1}, h_t) \tag{23.7.18}$$

for arbitrarily defined potentials $\phi_t(h_{t-1}, h_t)$*.*

1. *First consider*

$$\sum_{h_1,\ldots,h_T} \prod_{t=2}^{T} \phi_t(h_t, h_{t-1}) \tag{23.7.19}$$

Show that how the summation over h_1 may be pushed inside the product and that the result of the maximisation can be interpreted as a message

$$\alpha_{1\to 2}(h_2) = \sum_{h_1} \phi_2(h_1, h_2) \tag{23.7.20}$$

2. *Derive the recursion*

$$\alpha_{t-1\to t}(h_t) = \sum_{h_{t-1}} \phi_t(h_{t-1}, h_t)\alpha_{t-2\to t-1}(h_{t-1}) \tag{23.7.21}$$

3. *Similarly, show that one can push the summation of h_T inside the product to define*

$$\beta_{T-1\leftarrow T}(h_{T-1}) = \sum_{h_T} \phi_T(h_{T-1}, h_T) \tag{23.7.22}$$

and that by pushing in h_{T-1} etc. *one can define messages*

$$\beta_{t\leftarrow t+1}(h_t) = \sum_{h_{t+1}} \phi_{t+1}(h_t, h_{t+1})\beta_{t+1\leftarrow t+2}(h_{t+1}) \tag{23.7.23}$$

4. *Show that*

$$p(h_t, h_{t-1}) \propto \sum_{h_{t+1}} \alpha_{t-2\to t-1}(h_{t-1})\phi(h_{t-1}, h_t)\beta_{t\leftarrow t+1}(h_t) \tag{23.7.24}$$

Exercise 228. *A second order HMM is defined as*

$$p^{HMM2}(h_{1:T}, v_{1:T}) = p(h_1)p(v_1|h_1)p(h_2|h_1)p(v_2|h_2)\prod_{t=3}^{T} p(h_t|h_{t-1}, h_{t-2})p(v_t|h_t) \tag{23.7.25}$$

Following a similar approach to the first order HMM, derive explicitly a message passing algorithm to compute the most likely joint state

$$\operatorname*{argmax}_{h_{1:T}} p^{HMM2}(h_{1:T}|v_{1:T}) \tag{23.7.26}$$

Exercise 229. *Since the likelihood of the HMM can be computed using filtering only, in principle we do not need smoothing to maximise the likelihood (contrary to the EM approach). Explain how to compute the likelihood gradient by the use of filtered information alone (*i.e. *using only a forward pass).*

Exercise 230. *Derive the EM updates for fitting a HMM with an emission distribution given by a mixture of multi-variate Gaussians.*

Exercise 231. *Consider the HMM defined on hidden variables $\mathcal{H} = \{h_1, \ldots, h_T\}$ and observations $\mathcal{V} = \{v_1, \ldots, v_T\}$*

$$p(\mathcal{V}, \mathcal{H}) = p(h_1)p(v_1|h_1)\prod_{t=2}^{T} p(h_t|h_{t-1})p(v_t|h_t) \tag{23.7.27}$$

Show that the posterior $p(\mathcal{H}|\mathcal{V})$ is a Markov chain

$$p(\mathcal{H}|\mathcal{V}) = \tilde{p}(h_1)\prod_{t=2}^{T} \tilde{p}(h_t|h_{t-1}) \tag{23.7.28}$$

where $\tilde{p}(h_t|h_{t-1})$ and $\tilde{p}(h_1)$ are suitably defined distributions.

Exercise 232. *For training a HMM with a Gaussian mixture emission (the HMM-GMM model) in section(23.3.3), derive the following EM update formulae for the means and covariances:*

$$\boldsymbol{\mu}_{k,h}^{new} = \sum_{n=1}^{N}\sum_{t=1}^{T} \rho_{k,h}(t,n)\mathbf{v}_t^n \tag{23.7.29}$$

and

$$\boldsymbol{\mu}_{k,h}^{new} = \sum_{n=1}^{N}\sum_{t=1}^{T} \rho_{k,h}(t,n)\left(\mathbf{v}_t^n - \boldsymbol{\mu}_{k,h}\right)\left(\mathbf{v}_t^n - \boldsymbol{\mu}_{k,h}\right)^{\mathsf{T}} \tag{23.7.30}$$

where

$$\rho_{k,h}(t,n) = \frac{q(k_t = k|h_t^n = h)q(h_t^n = h|v_{1:T}^n)}{\sum_n \sum_t q(k_t = k|h_t^n = h)q(h_t^n = h|v_{1:T}^n)} \tag{23.7.31}$$

Exercise 233. *Consider the HMM duration model defined by equation (23.4.2) and equation (23.4.1) with emission distribution $p(v_t|h_t)$. Our interest is to derive a recursion for the filtered distribution*

$$\alpha_t(h_t, c_t) \equiv p(h_t, c_t, v_{1:t}) \tag{23.7.32}$$

1. *Show that :*

$$\alpha_t(h_t, c_t) = p(v_t|h_t) \sum_{h_{t-1},c_{t-1}} p(h_t|h_{t-1}, c_t)p(c_t|c_{t-1})\alpha_{t-1}(h_{t-1}, c_{t-1}) \tag{23.7.33}$$

2. *Using this derive*

$$\frac{\alpha_t(h_t, c_t)}{p(v_t|h_t)} = \sum_{h_{t-1}} p(h_t|h_{t-1}, c)p(c_t|c_{t-1} = 1)\alpha_{t-1}(h_{t-1}, c_{t-1} = 1) + \sum_{h_{t-1}} p(h_t|h_{t-1}, c) \sum_{c_{t-1}=2}^{D_{max}} p(c|c_{t-1})\alpha_{t-1}(h_{t-1}, c_{t-1}) \tag{23.7.34}$$

3. *Show that the right hand side of the above can be written as*

$$\sum_{h_{t-1}} p(h_t|h_{t-1}, c_t = c)p(c_t = c|c_{t-1} = 1)\alpha_{t-1}(h_{t-1}, 1) + \mathbb{I}\left[c \neq D_{max}\right] \sum_{h_{t-1}} p(h_t|h_{t-1}, c)\alpha_{t-1}(h_{t-1}, c + 1) \tag{23.7.35}$$

4. *Show that the recursion for α is then given by*

$$\alpha_t(h, 1) = p(v_t|h_t = h)p_{dur}(1) \sum_{h_{t-1}} p_{tran}(h|h_{t-1})\alpha_{t-1}(h_{t-1}, 1) + \mathbb{I}\left[D_{max} \neq 1\right] p(v_t|h_t = h) \sum_{h_{t-1}} p_{tran}(h_t|h_{t-1})\alpha_{t-1}(h_{t-1}, 2) \tag{23.7.36}$$

   and for $c > 1$

$$\alpha_t(h, c) = p(v_t|h_t = h)\left\{p_{dur}(c)\alpha_{t-1}(h, 1) + \mathbb{I}\left[c \neq D_{max}\right]\alpha_{t-1}(h, c + 1)\right\} \tag{23.7.37}$$

5. *Explain why the computational complexity of filtered inference in the duration model is $O\left(TH^2D_{max}\right)$.*

6. *Derive an efficient smoothing algorithm for this duration model.*