Discrete Latent State Dynamical Models (Chapter 23)[1]

David Barber

(adapted by Miguel Palomino)

[1] These slides accompany the book *Bayesian Reasoning and Machine Learning*. The book and demos can be downloaded from www.cs.ucl.ac.uk/staff/D.Barber/brml. Feedback and corrections are also available on the site. Feel free to adapt these slides for your own purposes, but please include a link the above website.

Time Series

A time series is an ordered sequence:

$$x_{a:b} = \{x_a, x_{a+1}, \ldots, x_b\}$$

So that one can consider the ‘past’ and ‘future’ in the sequence. The x can be either discrete or continuous.

Biology

Gene sequences. Emphasis is on understanding sequences, filling in missing values, clustering sequences, detecting patterns. Hidden Markov Models are one of the key tools in this area.

Finance

Price movement prediction.

Planning

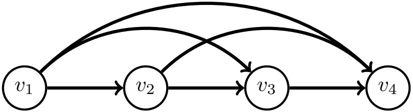
Forecasting – eg how many newspapers to deliver to retailers.

Markov Models

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

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

with the convention $p(v_t|v_{1:t-1}) = p(v_1)$ for $t = 1$.



Independence assumptions

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.

Markov Chain

Only the recent past is relevant:

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

where $L \geq 1$ is the order of the Markov chain

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

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

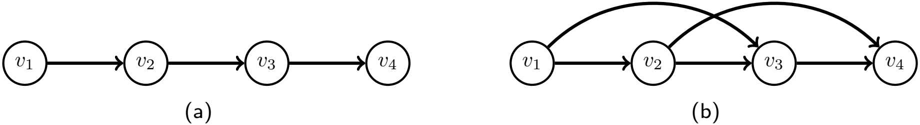


Figure: **(a)**: First order Markov chain. **(b)**: Second order Markov chain.

Fitting Markov models

Single series

Fitting a first-order stationary Markov chain by Maximum Likelihood corresponds to setting the transitions by counting the number of observed transitions in the sequence:

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

Multiple series

For a set of time series, $v^n_{1:T_n}, n = 1, \ldots, N$, the transition is given by counting all transitions across time and datapoints. The Maximum Likelihood setting for the initial first timestep distribution is $p(v_1 = i) \propto \sum_n \mathbb{I}\left[v^n_1 = i\right]$.

Hidden Markov Models

The 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

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

For a stationary HMM the transition $p(h_t|h_{t-1})$ and emission $p(v_t|h_t)$ distributions are constant through time.

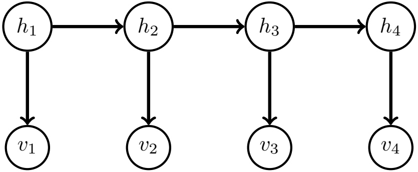


Figure: A first order hidden Markov model with 'hidden' variables $\mathrm{dom}(h_t) = \{1, \ldots, H\}$, $t = 1 : T$. The 'visible' variables v_t can be either discrete or continuous.

HMM parameters

Transition Distribution

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

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

and an initial distribution

$$a_i = p(h_1 = i).$$

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

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

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

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)	$\underset{h_{1:T}}{\operatorname{argmax}}\, p(h_{1:T} \| v_{1:T})$

For prediction, one is also often interested in $p(v_t|v_{1:s})$ for $t > s$.

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

$$\begin{aligned} p(h_t, v_{1:t}) &= \sum_{h_{t-1}} p(h_t, h_{t-1}, v_{1:t-1}, v_t) \\ &= \sum_{h_{t-1}} p(v_t|\cancel{v_{1:t-1}}, h_t, \cancel{h_{t-1}})p(h_t|\cancel{v_{1:t-1}}, h_{t-1})p(v_{1:t-1}, h_{t-1}) \\ &= \sum_{h_{t-1}} p(v_t|h_t)p(h_t|h_{t-1})p(h_{t-1}, v_{1:t-1}) \end{aligned}$$

Hence if we define $\alpha(h_t) \equiv p(h_t, v_{1:t})$ the above gives the α-recursion

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

with

$$\alpha(h_1) = p(h_1, v_1) = p(v_1|h_1)p(h_1)$$

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

Normalisation gives the filtered posterior

$$p(h_t|v_{1:t}) = \frac{p(h_t, v_{1:t})}{p(v_{1:t})} \propto \alpha(h_t)$$

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

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

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

One way to compute the smoothed quantity is to consider how h_t partitions the series into the past and future:

$$\begin{aligned} p(h_t, v_{1:T}) &= p(h_t, v_{1:t}, v_{t+1:T}) \\ &= \underbrace{p(h_t, v_{1:t})}_{\text{past}} \underbrace{p(v_{t+1:T}|h_t, \cancel{v_{1:t}})}_{\text{future}} = \alpha(h_t)\beta(h_t) \end{aligned}$$

Forward

The term $\alpha(h_t)$ is obtained from the 'forward' α recursion.

Backward

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.

The β recursion

$$
\begin{aligned}
p(v_{t:T}|h_{t-1}) &= \sum_{h_t} p(v_t, v_{t+1:T}, h_t|h_{t-1}) \\
&= \sum_{h_t} p(v_t|\cancel{v_{t+1:T}}, h_t, \cancel{h_{t-1}})p(v_{t+1:T}, h_t|h_{t-1}) \\
&= \sum_{h_t} p(v_t|h_t)p(v_{t+1:T}|h_t, \cancel{h_{t-1}})p(h_t|h_{t-1})
\end{aligned}
$$

Defining $\beta(h_t) \equiv p(v_{t+1:T}|h_t)$ gives the β-recursion

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

with $\beta(h_T) = 1$. The smoothed posterior is then given by

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

Together the $\alpha - \beta$ recursions are called the forward-backward algorithm.

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

To implement the EM algorithm for learning, we require terms such as $p(h_t, h_{t+1}|v_{1:T})$.

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

Rearranging, we therefore have

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

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 of

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

Consider

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

The message $\mu(h_{T-1})$ conveys information from the end of the chain to the penultimate timestep.

Most likely joint state

We can continue in this manner, defining the recursion

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

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

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

Once computed, backtracking gives

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

Prediction

Predicting the future hidden variable

$$p(h_{t+1}|v_{1:t}) = \sum_{h_t} p(h_{t+1}|h_t) \underbrace{p(h_t|v_{1:t})}_{filtering}$$

Predicting the future observation

The one-step ahead predictive distribution is given by

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

Burglar

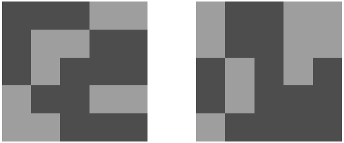
The nightmare scenario

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 is from listening to his movements.

Your calculation

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. Similarly you know for each position the probability that someone will bump into something in the dark. 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.

Burglar



(a) ‘Creaks’ (b) ‘Bumps’

Figure: 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.

Based on a series of bump/no bump and creak/no creak information 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 and we form a new visible variable with 4 states using

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

Burglar

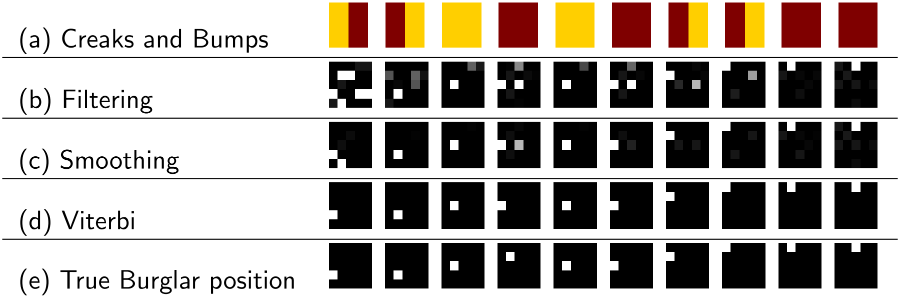


Figure: Localising the burglar through time for 10 time steps. **(a)**: Each panel represents the visible information $v_t = \left(v_t^{creak}, v_t^{bump}\right)$, where $v_t^{creak} = 1$ means that there was a 'creak in the floorboard' ($v_t^{creak} = 2$ otherwise) and $v_t^{bump} = 1$ meaning 'bumped into something'. **(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.

Learning HMMs

Given a set of data $\mathcal{V} = \left\{\mathbf{v}^1, \ldots, \mathbf{v}^N\right\}$ 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 generated $\mathcal{V}$. 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. The EM algorithm is also straightforward to implement in this case and leads to closed form expressions for the M-step.

EM algorithm

The application of EM to the HMM model is called the Baum-Welch algorithm.

M-step

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

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

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

Similarly,

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

which is the number of times that a transition from hidden state i to hidden state i' occurs, averaged over all times and training sequences. Finally,

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

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.

Parameter initialisation

The EM algorithm converges to a local maximum 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. 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.

Message passing inference

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.

Learning

For learning, however, the emission normalisation constant is required since this is dependent on the parameters of the model.

Mixture emission

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

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

where k_t is a discrete summation variable.

EM

For learning, it is useful to consider the k_t as additional latent variables, and then apply the standard EM algorithm.

The HMM-GMM

A common continuous observation mixture emission model component is a Gaussian

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

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. These models are common in tracking applications, in particular in speech recognition.

Update 2024

For speech recognition, HMMs with Gaussian mixture emissions have been largely superseded by deep learning approaches. However, they are still relevant in tracking applications, bioinformatics applications, finance, and quality control and anomaly detection in manufacturing.

Discriminative training

HMMs can be used for supervised learning of sequences. That is, for each sequence $v^n_{1:T}$, we have a corresponding class label c^n. For example, we might associate 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

$$p(c^*|v^*_{1:T}) = \frac{p(v^*_{1:T}|c^*)p(c^*)}{\sum_{c'=1}^{C} p(v^*_{1:T}|c')p(c')}$$

If the data are 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 the decision boundary.

Discriminative training

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

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

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

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

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.