Machine Learning for OR & FE

Hidden Markov Models

Martin Haugh

Department of Industrial Engineering and Operations Research
Columbia University
Email: martin.b.haugh@gmail.com

Additional References: David Barber's *Bayesian Reasoning and Machine Learning*

Outline

Hidden Markov Models

Hidden Markov Models (HMMs) are a rich class of models that have many applications including:

1. Target tracking and localization
2. Time-series analysis
3. Natural language processing and part-of-speech recognition
4. Speech recognition
5. Handwriting recognition
6. Stochastic control
7. Gene prediction
8. Protein folding
9. And many more . . .

HMMs are also the most important and commonly used class of graphical models

- and many of the algorithms that are used for HMMs can be adapted for more general use with graphical models.

HMMs are closely related to (non-linear) filtering problems and signal processing.

Hidden Markov Models

A HMM defines a Markov chain on data, $h_1, h_2, \ldots$, that is **hidden**. The goal is to to categorize this hidden data based on noisy or **visible** observations, $v_1, v_2, \ldots$.

Individual observations may be difficult to categorize by themselves:

But the task becomes much easier when the observations are taken in the context of the entire visible sequence:

(This example is taken from Ben Taskar's website at
`https://alliance.seas.upenn.edu/~cis520/wiki/index.php?n=Lectures.HMMs#toc7`)

Graphical Representation of HMMs

We often represent HMMs using the graphical model representation

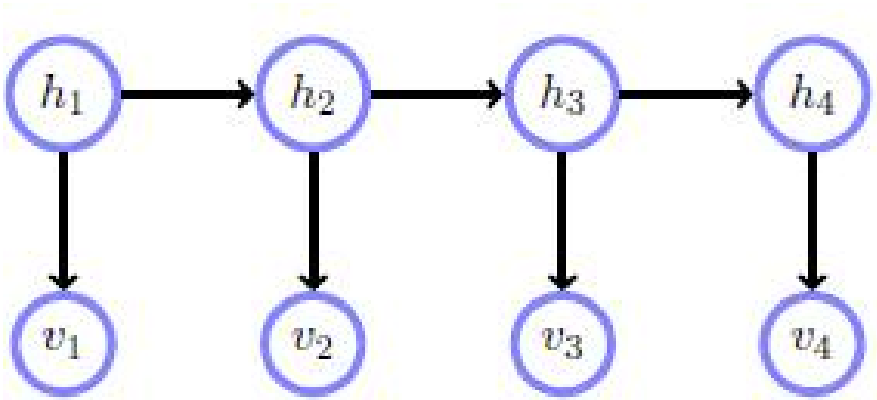


Figure 23.4 from Barber: A first order hidden Markov model with 'hidden' variables $\text{dom}(h_t) = \{1, \ldots, H\}$, $t = 1 : T$. The 'visible' variables v_t can be either discrete or continuous.

A HMM is defined by:

1. The initial distribution, $\mathsf{P}(h_1)$.
2. The transition distribution, $\mathsf{P}(h_t \,|\, h_{t-1})$, of the underlying Markov chain.
3. The emission distribution, $\mathsf{P}(v_t \,|\, h_t)$.

The HMM model is therefore very simple

- but it has been very successful in many application domains.

Inference for Hidden Markov Models

The main inference problems are:

1. The filtering problem solves for $\mathsf{P}\left(h_t \mid v_{1:t}\right)$
   - inferring the present.
2. The smoothing problem computes $\mathsf{P}\left(h_t \mid v_{1:T}\right)$ for $t < T$
   - inferring the past.
3. The prediction problem solves for $\mathsf{P}\left(h_t \mid v_{1:s}\right)$, for $t > s$
   - inferring the future.
4. The likelihood problem solves for $\mathsf{P}\left(v_{1:T}\right)$.
5. The most likely hidden path(s) problem solves for

$$\operatorname*{argmax}_{h_{1:T}} \mathsf{P}\left(h_{1:T} \mid v_{1:T}\right) \equiv \operatorname*{argmax}_{h_{1:T}} \mathsf{P}\left(h_{1:T},\, v_{1:T}\right).$$

These problems can be solved **efficiently** using dynamic programming techniques.

Note also that in addressing these inference problems, the particular form of $\mathsf{P}(v_t \mid h_t)$ is not important.

The Filtering Problem

We first compute $\alpha(h_t) := \mathsf{P}(h_t, v_{1:t})$

- gives the un-normalized filtered posterior distribution
- can easily normalize to compute $\mathsf{P}(h_t \mid v_{1:t}) \propto \alpha(h_t)$.

We begin with $\alpha(h_1) := \mathsf{P}(v_1|h_1)\mathsf{P}(h_1)$. Now note that

$$\begin{aligned}
\alpha(h_t) &= \sum_{h_{t-1}} \mathsf{P}(h_t, h_{t-1}, v_{1:t-1}, v_t) \\
&= \sum_{h_{t-1}} \mathsf{P}(v_t|h_t, h_{t-1}, v_{1:t-1})\ \mathsf{P}(h_t|h_{t-1}, v_{1:t-1})\ \mathsf{P}(h_{t-1}, v_{1:t-1}) \\
&= \sum_{h_{t-1}} \mathsf{P}(v_t|h_t)\ \mathsf{P}(h_t|h_{t-1})\ \mathsf{P}(h_{t-1}, v_{1:t-1}) \\
&= \underbrace{\mathsf{P}(v_t|h_t)}_{\text{corrector}} \underbrace{\sum_{h_{t-1}} \mathsf{P}(h_t|h_{t-1})\ \alpha(h_{t-1})}_{\text{predictor}}. \qquad (1)
\end{aligned}$$

Can therefore solve the filtering problem in $O(T)$ time.

The Smoothing Problem

Now compute $\beta(h_t) := \mathsf{P}(v_{t+1:T}|h_t)$ with the understanding that $\beta(h_T) = 1$:

$$
\begin{aligned}
\beta(h_t) &= \sum_{h_{t+1}} \mathsf{P}(v_{t+1}, v_{t+2:T}, h_{t+1}|h_t) \\
&= \sum_{h_{t+1}} \mathsf{P}(v_{t+1}|v_{t+2:T}, h_{t+1}, h_t)\, \mathsf{P}(v_{t+2:T}, h_{t+1}|h_t) \\
&= \sum_{h_{t+1}} \mathsf{P}(v_{t+1}|v_{t+2:T}, h_{t+1}, h_t)\, \mathsf{P}(v_{t+2:T}|h_{t+1}, h_t)\, \mathsf{P}(h_{t+1}|h_t) \\
&= \sum_{h_{t+1}} \mathsf{P}(v_{t+1}|h_{t+1})\, \mathsf{P}(h_{t+1}|h_t)\, \beta(h_{t+1}). \qquad (2)
\end{aligned}
$$

The Smoothing Problem

Now note that

$$\begin{aligned} \mathsf{P}\left(h_t, v_{1:T}\right) &= \mathsf{P}\left(h_t, v_{1:t}\right) \mathsf{P}\left(v_{t+1:T} | h_t, v_{1:t}\right) \\ &= \mathsf{P}\left(h_t, v_{1:t}\right) \mathsf{P}\left(v_{t+1:T} | h_t\right) \\ &= \alpha(h_t)\beta(h_t). \end{aligned}$$

We therefore obtain (why?)

$$\mathsf{P}\left(h_t | v_{1:T}\right) = \frac{\alpha(h_t)\beta(h_t)}{\sum_{h_t} \alpha(h_t)\beta(h_t)} \tag{3}$$

which solves the smoothing problem.

The α-β recursions are often called the forward-backward recursion.

Prediction and the Likelihood

The one-step ahead predictive distribution is given by

$$\begin{aligned}
\mathsf{P}(v_{t+1}|v_{1:t}) &= \sum_{h_t,h_{t+1}} \mathsf{P}\left(h_t, h_{t+1}, v_{t+1}|v_{1:t}\right) \\
&= \sum_{h_t,h_{t+1}} \mathsf{P}\left(v_{t+1}|h_t, h_{t+1}, v_{1:t}\right) \mathsf{P}\left(h_{t+1}|h_t, v_{1:t}\right) \mathsf{P}\left(h_t|v_{1:t}\right) \\
&= \sum_{h_t,h_{t+1}} \mathsf{P}\left(v_{t+1}|h_{t+1}\right) \mathsf{P}(h_{t+1}|h_t) \mathsf{P}(h_t|v_{1:t})
\end{aligned}$$

which is easy to calculate given the filtering distribution, $\mathsf{P}(h_t|v_{1:t})$.

The likelihood $\mathsf{P}(v_{1:T})$ can be calculated in many ways including

$$\begin{aligned}
\mathsf{P}(v_{1:T}) &= \sum_{h_T} \mathsf{P}(h_T, v_{1:T}) \\
&= \sum_{h_T} \alpha(h_T).
\end{aligned}$$

Computing the Pairwise Marginal $\mathsf{P}\left(h_t, h_{t+1} | v_{1:T}\right)$

Can compute $\mathsf{P}\left(h_t, h_{t+1} | v_{1:T}\right)$ by noting that

$$\begin{aligned}
\mathsf{P}\left(h_t, h_{t+1} | v_{1:T}\right) &\propto \mathsf{P}\left(v_{1:t}, v_{t+1}, v_{t+2:T}, h_{t+1}, h_t\right) \\
&= \mathsf{P}\left(v_{t+2:T} | v_{1:t}, v_{t+1}, h_{t+1}, h_t\right) \mathsf{P}\left(v_{1:t}, v_{t+1}, h_{t+1}, h_t\right) \\
&= \mathsf{P}\left(v_{t+2:T} | h_{t+1}\right) \mathsf{P}\left(v_{t+1} | v_{1:t}, h_{t+1}, h_t\right) \mathsf{P}\left(v_{1:t}, h_{t+1}, h_t\right) \\
&= \mathsf{P}\left(v_{t+2:T} | h_{t+1}\right) \mathsf{P}\left(v_{t+1} | h_{t+1}\right) \mathsf{P}\left(h_{t+1} | v_{1:t}, h_t\right) \mathsf{P}\left(v_{1:t}, h_t\right) \\
&= \mathsf{P}\left(v_{t+2:T} | h_{t+1}\right) \mathsf{P}\left(v_{t+1} | h_{t+1}\right) \mathsf{P}\left(h_{t+1} | h_t\right) \mathsf{P}\left(v_{1:t}, h_t\right). \quad (4)
\end{aligned}$$

Can rearrange (4) to obtain

$$\mathsf{P}\left(h_t, h_{t+1} | v_{1:T}\right) \propto \alpha(h_t) \mathsf{P}\left(v_{t+1} | h_{t+1}\right) \mathsf{P}\left(h_{t+1} | h_t\right) \beta(h_{t+1}) \quad (5)$$

So $\mathsf{P}\left(h_t, h_{t+1} | v_{1:T}\right)$ easy to compute once forward-backward recursions have been completed

- pairwise marginals are used by the EM algorithm for **learning** the HMM
- in this context the EM algorithm is called the Baum-Welch algorithm.

Sampling from the Posterior

Sometimes we would like to sample from the posterior $\mathsf{P}\left(h_{1:T}|v_{1:T}\right)$.

One straightforward way to do is by first noting that

$$\begin{aligned}\mathsf{P}\left(h_{1:T}|v_{1:T}\right) &= \mathsf{P}\left(h_1|h_{2:T},v_{1:T}\right)\ \ldots\ \mathsf{P}\left(h_{T-1}|h_T,v_{1:T}\right)\mathsf{P}\left(h_T|v_{1:T}\right)\\ &= \mathsf{P}\left(h_1|h_2,v_{1:T}\right)\ \ldots\ \mathsf{P}\left(h_{T-1}|h_T,v_{1:T}\right)\mathsf{P}\left(h_T|v_{1:T}\right) \qquad (6)\end{aligned}$$

So can sample sequentially by:

- First drawing h_T from $\mathsf{P}\left(h_T|v_{1:T}\right)$
- And then noting that for any $t < T$ we have

$$\begin{aligned}\mathsf{P}\left(h_t|h_{t+1},v_{1:T}\right) &\propto \mathsf{P}\left(h_t,h_{t+1}|v_{1:T}\right)\\ &\propto \alpha(h_t)\mathsf{P}\left(h_{t+1}|h_t\right) \quad \text{by } (5)\end{aligned}$$

  from which it is easy to sample.

This sequential sampling process is known as forward-filtering-backward sampling.

Computing the Most Likely Hidden Path

The most likely hidden path problem is found by solving

$$
\begin{aligned}
\max_{h_{1:T}} \mathsf{P}\left(h_{1:T} \,|\, v_{1:T}\right) \quad &\equiv \quad \max_{h_{1:T}} \mathsf{P}\left(h_{1:T},\, v_{1:T}\right) \\
&= \quad \max_{h_{1:T}} \prod_{t=1}^{T} \mathsf{P}\left(v_t|h_t\right)\, \mathsf{P}\left(h_t|h_{t-1}\right) \\
&= \quad \max_{h_{1:T-1}} \left\{ \prod_{t=1}^{T-1} \mathsf{P}\left(v_t|h_t\right)\, \mathsf{P}\left(h_t|h_{t-1}\right) \right\} \underbrace{\max_{h_T}\; \mathsf{P}\left(v_T|h_T\right)\, \mathsf{P}\left(h_T|h_{T-1}\right)}_{\mu(h_{T-1})} \\
&= \quad \max_{h_{1:T-2}} \left\{ \prod_{t=1}^{T-2} \mathsf{P}\left(v_t|h_t\right)\, \mathsf{P}\left(h_t|h_{t-1}\right) \right\} \\
&\qquad\qquad \times \underbrace{\max_{h_{T-1}}\; \mathsf{P}\left(v_{T-1}|h_{T-1}\right)\, \mathsf{P}\left(h_{T-1}|h_{T-2}\right) \mu(h_{T-1})}_{\mu(h_{T-2})} \\
&= \quad \ldots
\end{aligned}
$$

The Viterbi Algorithm

We can therefore find the most likely hidden path by using the recursion

$$\mu(h_{t-1}) \;:=\; \max_{h_t} \mathsf{P}\left(v_t|h_t\right)\, \mathsf{P}\left(h_t|h_{t-1}\right) \mu(h_t) \tag{7}$$

to obtain $\mu(h_1), \ldots, \mu(h_{T-1})$.

Once we've solve for $\mu(h_1)$ we can backtrack to obtain the most likely hidden path. We get

$$h_1^* = \operatorname*{argmax}_{h_1} \mathsf{P}\left(v_1|h_1\right) \mathsf{P}\left(h_1\right) \mu(h_1)$$

and for $t = 2, \ldots, T$ we find

$$h_t^* = \operatorname*{argmax}_{h_t} \mathsf{P}\left(v_t|h_t\right) \mathsf{P}\left(h_t|h_{t-1}^*\right) \mu(h_t)$$

where we have defined $\mu(h_T) \equiv 1$.

This algorithm is called the Viterbi algorithm

- can be easily generalized (at the cost of some tedious 'book-keeping') to find the N most likely paths.

Very similar algorithms are used more generally in graphical models.

Application #1: Localization and Target Tracking

Example 23.3 from Barber:

1. You are asleep upstairs and are suddenly woken by a burglar downstairs. You want to figure out where he is by listening to his movements.
2. So you partition the ground floor into a 5×5 grid.
3. For each grid position you know the probability of: (i) bumping into something and (ii) the floorboard at that position creaking
   – these probabilities are assumed to be independent.
4. You also assume that the burglar will only move 1 grid-square (forwards, backwards, left or right) at a time and that these transition probabilities are known.
5. The burglar leaves at time $T = 10$ and in order to help the police, you wish to construct the smoothed distribution $\mathsf{P}(h_t \,|\, v_{1:T})$ where:
   (a) h_t was the position of the burglar at time t and
   (b) $v_t = v_t^{\text{creak}} \otimes v_t^{\text{bump}} \in \{1,2\} \otimes \{1,2\}$ is the time t observation with $1 \equiv$ creak / bump and $2 \equiv$ no creak / no bump.

By assumption $\mathsf{P}(h_1)$, $\mathsf{P}(h_t \,|\, h_{t-1})$ and $\mathsf{P}(v_t \,|\, h_t) = \mathsf{P}(v_t^{\text{creak}} \,|\, h_t)\mathsf{P}(v_t^{\text{bump}} \,|\, h_t)$ are known for all t.

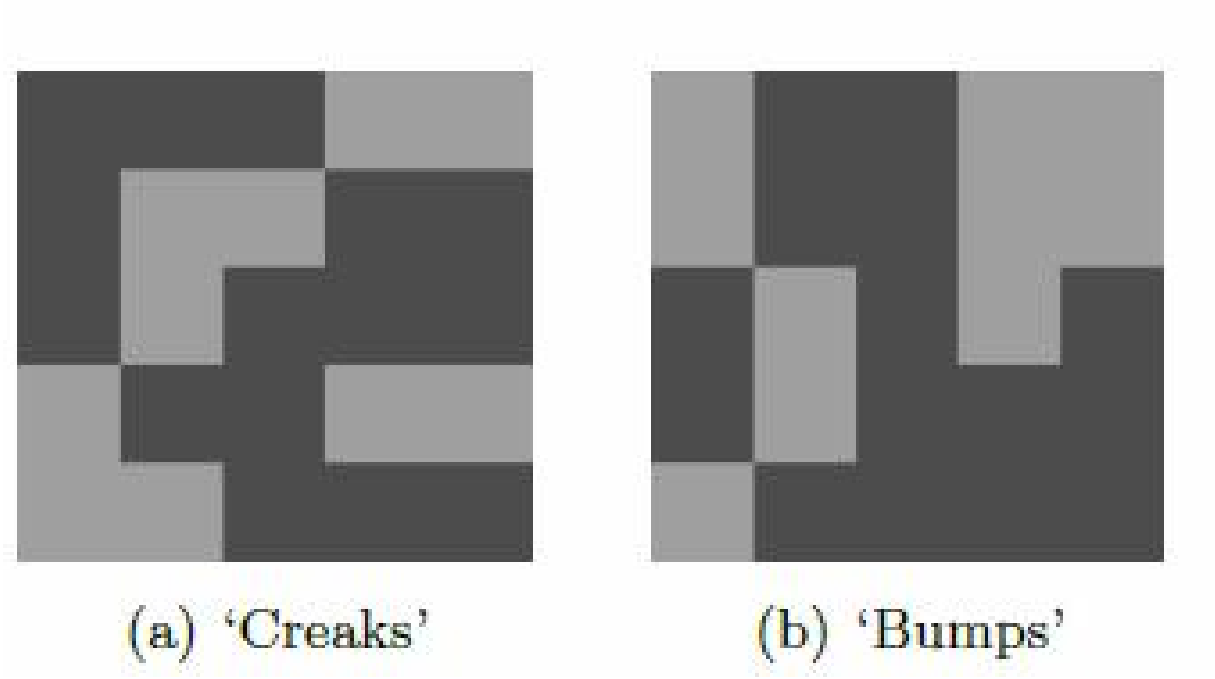


Figure 23.6 from Barber: Localising the burglar. The latent variable $h_t \in \{1, \ldots, 25\}$ denotes the positions, defined over the 5×5 grid of the ground floor of the house. (a): A representation of the probability that the 'floor will creak' at each of the 25 positions, $p(v^{\text{creak}} \,|\,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

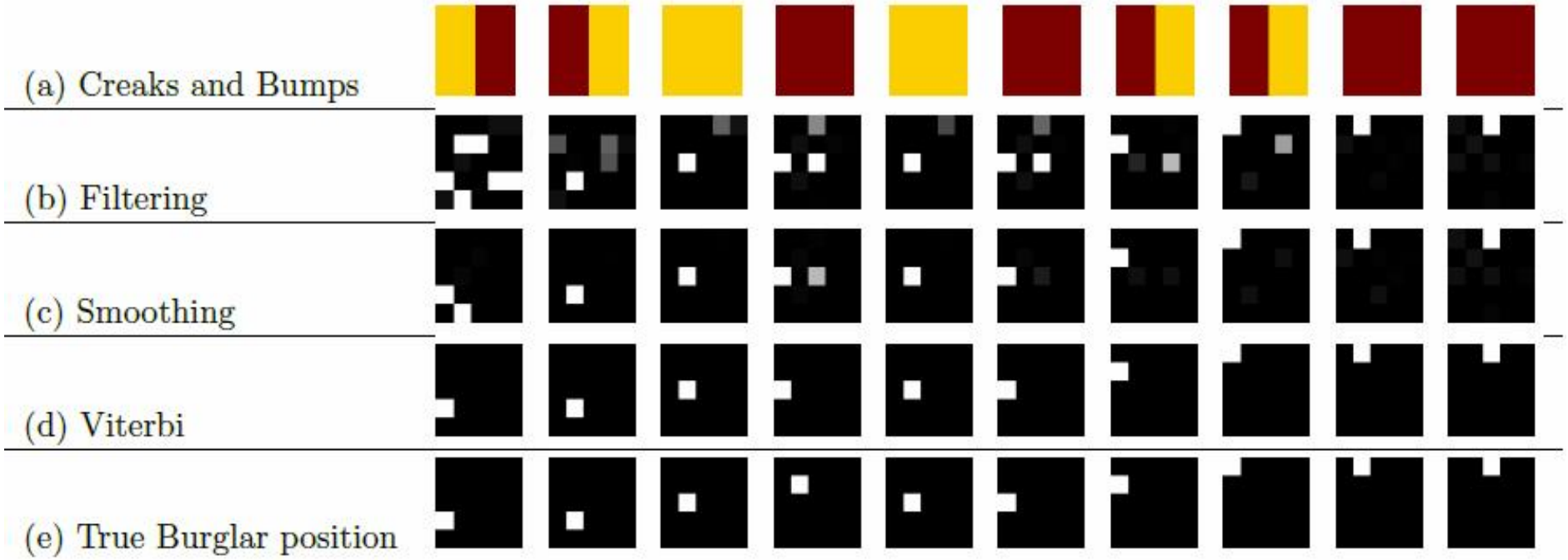


Figure 23.7 from Barber: Localising the burglar through time for 10 time steps. (a): Each panel represents the visible information $v_t = \left(v_t^{\text{creak}}, v_t^{\text{bump}}\right)$, where $v^{\text{creak}} = 1$ means that there was a 'creak in the floorboard' ($v^{\text{creak}} = 2$ otherwise) and $v^{\text{bump}} = 1$ 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^{creak} and the right half v_t^{bump}. The yellow shade represents the occurrence of a creak or bump, the red shade the absence. (b): The filtered distribution $p(h_t \mid v_{1:t})$ representing where we think the burglar is. (c): The smoothed distribution $p(h_t \mid v_{1:10})$ that represents the distribution of the burglar's position given that we know both the past and future observations. (d): The most likely (Viterbi) burglar path $\text{argmax}_{h_{1:10}}\, p(h_{1:10} \mid v_{1:10})$. (e): The actual path of the burglar.

Application #2: Stubby Fingers and NLP

Example 23.5 from Barber:

1. A "stubby fingers" typist hits either the correct key or a neighboring key every times he / she types.
2. Assume there are 27 keys: lower case 'a' to lower case 'z', and the space bar.
3. We don't know what key, h_t, the typist intended to hit at time t and only observe v_t, the actual key that was typed.
4. The emission distribution, $B_{ij} := \mathsf{P}(v = i \,|\, h = j)$, is easily estimated
   - and is depicted in Figure 23.10(b) from Barber.
5. Transition matrix, $A_{ij} := \mathsf{P}(h' = i \,|\, h = j)$, easily estimated from a database of letter-to-next-letter frequencies in English
   - and is depicted in Figure 23.10(a) from Barber.
6. Can simply assume that the initial distribution $\mathsf{P}(h_1)$ is uniform.

Question: Given a typed sequence "`kezrninh`" what is the most likely word / phrase that this corresponds to?

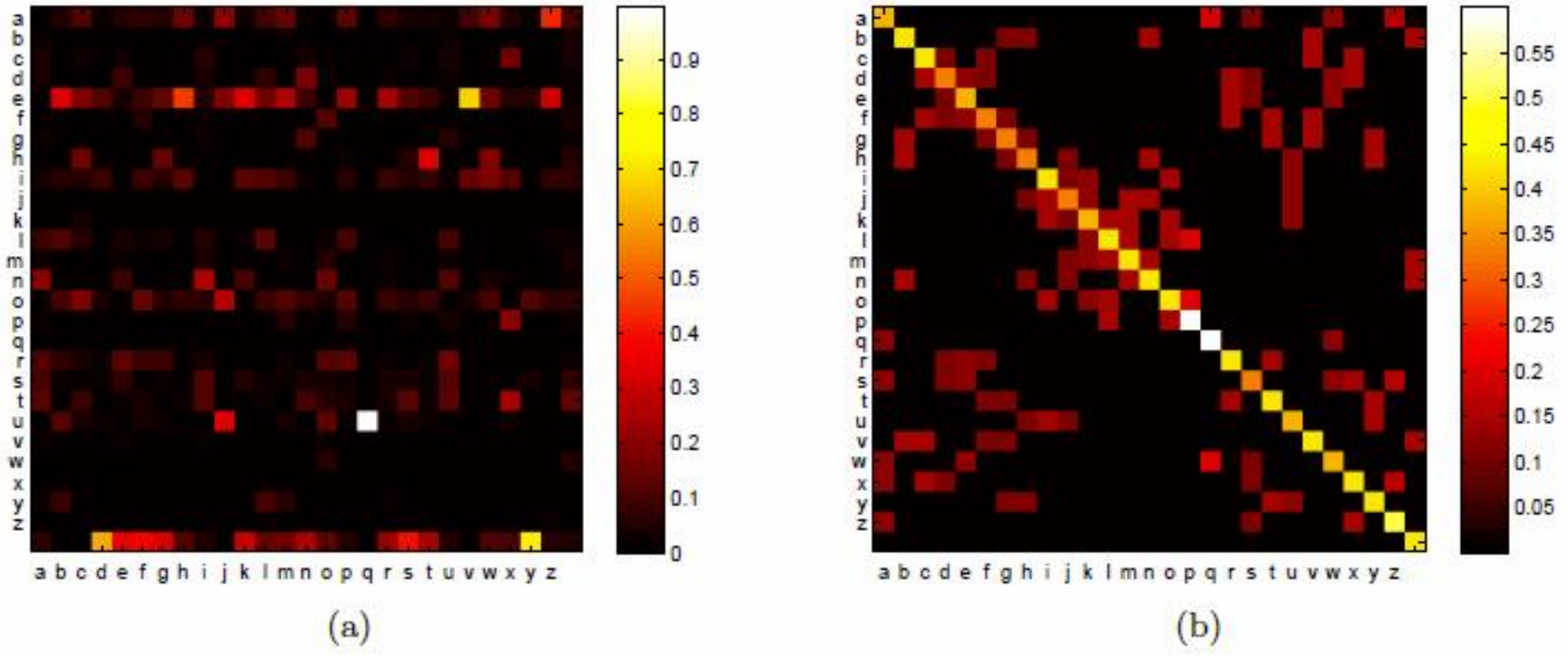


Figure 23.10 from Barber: (a): The letter-to-letter transition matrix for English $p(h' = i \mid h = j)$. (b): The letter emission matrix for a typist with 'stubby fingers' in which the key or a neighbour on the keyboard is likely to be hit.

Can answer this using a generalization of the Viterbi algorithm – the N-max-product algorithm – which finds the N most likely hidden paths

- and then compares these N most likely phrases, i.e. paths, with words or phrases from a standard English dictionary.

Application #3: Self-Localization

Consider a robot with an internal grid-based map of its environment.

For each location $h \in \{1, \ldots, H\}$ the robot "knows" the likely sensor readings it would obtain in that location.

The robot's goal is to move about and take sensor readings, and by comparing these readings to its internal map, allow it to estimate its location.

More specifically, the robot makes **intended** movements $m_{1:t}$

- but due to floor slippage etc, these movements aren't always successful
- one can view the intended movements as additional observed information.

Robot also gathers sensor information, $v_{1:t}$, from the unobserved locations, $h_{1:t}$.

Application #3: Self-Localization

We can model this problem according to

$$\mathsf{P}(v_{1:T}, m_{1:T}, h_{1:T}) \;=\; \prod_{t=1}^{T} \mathsf{P}(v_t \,|\, h_t)\mathsf{P}(h_t \,|\, h_{t-1}, m_{t-1})\mathsf{P}(m_t) \tag{8}$$

- where we have allowed for the possibility that the robot makes intended movements randomly.

Since $m_{1:t}$ is known to the robot we can rewrite (8) as a time-dependent HMM:

$$\mathsf{P}(v_{1:T}, h_{1:T}) \;=\; \prod_{t=1}^{T} \mathsf{P}(v_t \,|\, h_t)\mathsf{P}_t(h_t \,|\, h_{t-1}). \tag{9}$$

All our earlier inference algorithms still apply as long as we replace $\mathsf{P}(h_t \,|\, h_{t-1})$ with $\mathsf{P}_t(h_t \,|\, h_{t-1})$ everywhere.

An Example of Self-Localization: Robot Tracking

Example 23.4 from Barber:

1. A robot is moving around a circular corridor and at any time occupies one of S possible locations.
2. At each time t the robot stays where it is with probability ϵ and moves one space counter-clockwise with probability $1-\epsilon$
   - so no intended movements, $m_{1:t}$, here.

   Can represent this with a matrix $\mathbf{A}_{ij} = \mathsf{P}(h_t = j \,|\, h_{t-1} = i)$. **e.g.** if $S = 3$

$$\mathbf{A} \;=\; \epsilon \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \;+\; (1-\epsilon) \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

3. At each time t robot sensors measure its position and obtains correct location with prob. ω or a uniformly random location with prob. $1-\omega$.

   Can represent this with a matrix $\mathbf{B}_{ij} = \mathsf{P}(v_t = j \,|\, h_t = i)$. **e.g.** if $S = 3$ then

$$\mathbf{B} \;=\; \omega \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \;+\; \frac{(1-\omega)}{3} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

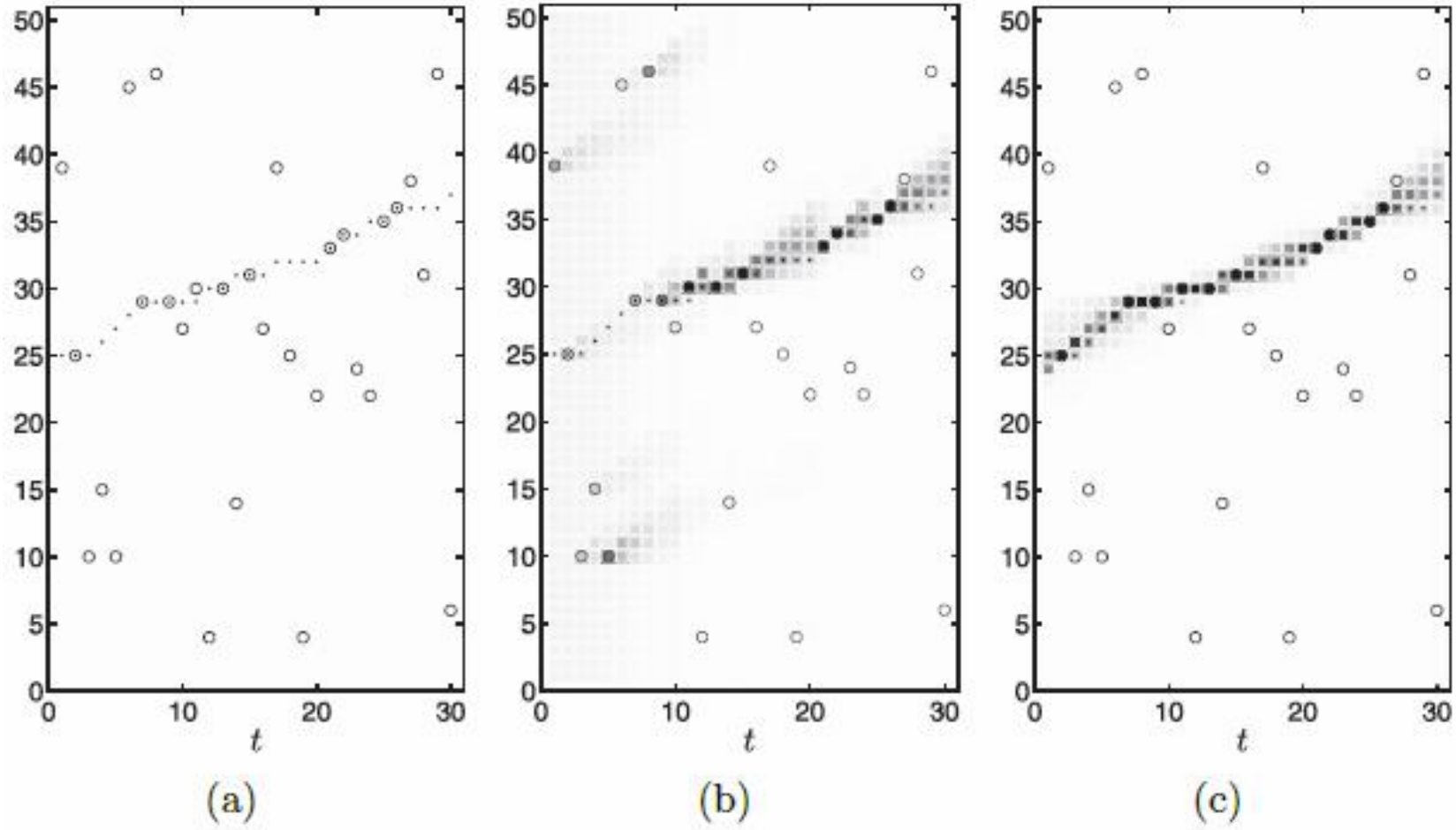


Figure 23.9 from Barber: Filtering and smoothing for robot tracking using a HMM with $S = 50$ and $\epsilon = .5$. (a): A realisation from the HMM example described in the text. The dots indicate the true latent locations of the robot, whilst the open circles indicate the noisy measured locations. (b): The squares indicate the filtering distribution at each time-step t, $p(h_t \mid v_{1:t})$. This probability is proportional to the grey level with black corresponding to 1 and white to 0. Note that the posterior for the first time-steps is multimodal, therefore the true position cannot be accurately estimated. (c): The squares indicate the smoothing distribution at each time-step t, $p(h_t \mid v_{1:T})$. Note that, for $t < T$, we estimate the position retrospectively and the uncertainty is significantly lower when compared to the filtered estimates.

Learning Hidden Markov Models

Training or learning, a HMM requires us to estimate $\boldsymbol{\theta}$ which consists of:

1. the initial distribution, $\mathrm{P}(h_1)$
2. the transition distribution, $\mathrm{P}(h' \,|\, h)$ for all h
3. the emission distribution, $\mathrm{P}(v \,|\, h)$ for all h

There are two possible situations to consider:

1. We have access to labeled training sequences $(v^i_{1:T_i}, h^i_{1:T_i})$ for $i = 1, \ldots, n$
   - in this case can estimate all probabilities using standard ML estimation
2. We only have access to the sequences $(v^i_{1:T_i})$ for $i = 1, \ldots, n$
   - in this case need to use EM algorithm which in the context of HMMs if often called Baum-Welch.

Will consider each case separately and assume for ease of exposition that we have a discrete state space with K hidden states and a discrete observation space with M possible observations

- but easy to adapt to Gaussian emission distributions etc.

Learning HMMs Given Labeled Sequences

When we are given labeled sequences the log-likelihood, l, satisfies

$$
\begin{aligned}
l &= \sum_{i=1}^{n} \log \mathsf{P}\left(v^i_{1:T_i}, h^i_{1:T_i}\right) \\
&= \sum_{i=1}^{n} \log \prod_{t=1}^{T_i} \mathsf{P}\left(v^i_t \,|\, h^i_t\right)\ \mathsf{P}\left(h^i_t \,|\, h^i_{t-1}\right) \\
&= \sum_{i=1}^{n} \sum_{t=1}^{T_i} \left(\log \mathsf{P}\left(v^i_t \,|\, h^i_t\right)\ +\ \log \mathsf{P}\left(h^i_t | h^i_{t-1}\right)\right) \\
&= \sum_{k=1}^{K} \sum_{m=1}^{M} \sum_{i=1}^{n} \sum_{t=1}^{T_i} 1_{\{h^i_t=k\}} 1_{\{v^i_t=m\}} \log \mathsf{P}\left(v^i_t = m \,|\, h^i_t = k\right)\ + \\
&\quad \sum_{k=1}^{K} \sum_{k'=1}^{K} \sum_{i=1}^{n} \sum_{t=1}^{T_i} 1_{\{h^i_t=k', h^i_{t-1}=k\}} \log \mathsf{P}\left(h^i_t = k' \,|\, h^i_{t-1} = k\right) \qquad (10)
\end{aligned}
$$

with the understanding that $\mathsf{P}\left(h^i_1 \,|\, h^i_0\right) \equiv \mathsf{P}(h^i_1)$.

Learning HMMs Given Labeled Sequences

Not surprisingly the ML estimation problem therefore decomposes and it is easy to see that the solution is

$$\hat{\mathsf{P}}(h_1 = h) = \frac{1}{n}\sum_{i=1}^{n} 1_{\{h_1^i = h\}} \tag{11}$$

$$\hat{\mathsf{P}}(h' \,|\, h) = \frac{\sum_{i=1}^{n}\sum_{t=2}^{T_i} 1_{\{h_{t-1}^i = h,\, h_t^i = h'\}}}{\sum_{i=1}^{n}\sum_{t=2}^{T_i} 1_{\{h_{t-1}^i = h\}}} \tag{12}$$

$$\hat{\mathsf{P}}(v \,|\, h) = \frac{\sum_{i=1}^{n}\sum_{t=1}^{T_i} 1_{\{h_t^i = h\}}\, 1_{\{v_t^i = v\}}}{\sum_{i=1}^{n}\sum_{t=1}^{T_i} 1_{\{h_t^i = h\}}} \tag{13}$$

The Baum-Welch (EM) Algorithm

Suppose now we are only given the unlabeled sequences, $(v^i_{1:T_i})$ for $i = 1, \ldots, n$.

We begin with an initial guess, $\boldsymbol{\theta}^0$ say, of the model parameters and iterate the following E- and M-steps:

E-Step: We need to compute $Q(\boldsymbol{\theta}, \boldsymbol{\theta}_0)$, the expected complete-data log-likelihood, conditional on $(v^i_{1:T_i})$ for $i = 1, \ldots, n$

- note that the expectation is computed using $\boldsymbol{\theta}^0$.

Using (10) we obtain

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta}_0) \quad = \quad & \sum_{k=1}^{K} \sum_{m=1}^{M} \sum_{i=1}^{n} \sum_{t=1}^{T_i} q(h^i_t = k) 1_{\{v^i_t = m\}} \log \mathsf{P}\left(v^i_t = m \,|\, h^i_t = k\right) \; + \\ & \sum_{k=1}^{K} \sum_{k'=1}^{K} \sum_{i=1}^{n} \sum_{t=1}^{T_i} q(h^i_t = k', h^i_{t-1} = k) \log \mathsf{P}\left(h^i_t = k' \,|\, h^i_{t-1} = k\right) \end{aligned} \tag{14}$$

where $q(\cdot)$ denotes the marginal (and pairwise marginal) computed using $\boldsymbol{\theta}_0$

- and we know how to compute $q(\cdot)$.

The Baum-Welch (EM) Algorithm

M-Step: In the M-step we maximize $Q(\boldsymbol{\theta}, \boldsymbol{\theta}_0)$ with respect to $\boldsymbol{\theta}$.

The solution is analogous to (11), (12) and (13) except that we must replace the indicator functions with $q(\cdot)$:

$$\hat{\mathsf{P}}(h_1 = h) = \frac{1}{n} \sum_{i=1}^{n} q(h_1^i = h) \tag{15}$$

$$\hat{\mathsf{P}}(h' \,|\, h) = \frac{\sum_{i=1}^{n} \sum_{t=2}^{T_i} q(h_{t-1}^i = h,\, h_t^i = h')}{\sum_{i=1}^{n} \sum_{t=2}^{T_i} q(h_{t-1}^i = h)} \tag{16}$$

$$\hat{\mathsf{P}}(v \,|\, h) = \frac{\sum_{i=1}^{n} \sum_{t=1}^{T_i} q(h_t^i = h)\, 1_{\{v_t^i = v\}}}{\sum_{i=1}^{n} \sum_{t=1}^{T_i} q(h_t^i = h)}. \tag{17}$$

We then iterate the E-step and M-step until convergence.

Appendix: Extensions of HMMs

There are many **immediate** and **tractable** extensions of HMMs including:

1. Explicit Duration HMMs
2. Input-Output HMMs
   - the self-localization application with intended movements $m_{1:T}$ is an example.
3. Dynamic Bayesian Networks.
4. Autoregressive HMMs
   – used to capture dependence between the observed variables.

All of these extensions are discussed in Section 23.4 of Barber and / or Section 13.2.6 of Bishop.

More generally, HMMs can be viewed as an example of graphical models

- where the problems of inference and learning reduce to finding efficient ways to do addition and multiplication of the appropriate probabilities.

Appendix: Linear-Gaussian Dynamical Systems

It is always the case that the hidden variables in a HMM are **discrete**

- this need not be the case for the emission variables or observations.

Can also consider dynamic models where the hidden variables are **continuous**

- the most well-known such model is the Linear-Gaussian model:

$$\begin{aligned} \mathbf{h}_t &= \mathbf{A}_t \mathbf{h}_{t-1} + \boldsymbol{\eta}_t^h, & \boldsymbol{\eta}_t^h &\sim \mathsf{N}\left(\bar{\mathbf{h}}_t, \boldsymbol{\Sigma}_t^h\right) \\ \mathbf{v}_t &= \mathbf{B}_t \mathbf{h}_t + \boldsymbol{\eta}_t^v, & \boldsymbol{\eta}_t^v &\sim \mathsf{N}\left(\bar{\mathbf{v}}_t, \boldsymbol{\Sigma}_t^v\right) \end{aligned}$$

  where the $\mathbf{h}_t$'s are hidden and the $\mathbf{v}_t$'s are observed.

Great advantage of the linear-Gaussian model is that it is also tractable

- not surprising given the Gaussian and linear assumptions.

Filtering and smoothing algorithms are as shown on next two slides.

Algorithm 24.1 LDS Forward Pass. Compute the filtered posteriors $p(\mathbf{h}_t|\mathbf{v}_{1:t}) \equiv \mathcal{N}(\mathbf{f}_t, \mathbf{F}_t)$ for a LDS with parameters $\theta_t = \{\mathbf{A}, \mathbf{B}, \boldsymbol{\Sigma}^h, \boldsymbol{\Sigma}^v, \bar{\mathbf{h}}, \bar{\mathbf{v}}\}_t$. The log-likelihood $L = \log p(\mathbf{v}_{1:T})$ is also returned.

$\{\mathbf{f}_1, \mathbf{F}_1, p_1\} = \text{LDSFORWARD}(\mathbf{0}, \mathbf{0}, \mathbf{v}_1; \theta_1)$
$L \leftarrow \log p_1$
for $t \leftarrow 2, T$ **do**
  $\{\mathbf{f}_t, \mathbf{F}_t, p_t\} = \text{LDSFORWARD}(\mathbf{f}_{t-1}, \mathbf{F}_{t-1}, \mathbf{v}_t; \theta_t)$
  $L \leftarrow L + \log p_t$
end for
function LDSFORWARD$(\mathbf{f}, \mathbf{F}, \mathbf{v}; \theta)$
  $\boldsymbol{\mu}_h \leftarrow \mathbf{A}\mathbf{f} + \bar{\mathbf{h}}, \quad \boldsymbol{\mu}_v \leftarrow \mathbf{B}\boldsymbol{\mu}_h + \bar{\mathbf{v}}$ ▷ Mean of $p(\mathbf{h}_t, \mathbf{v}_t|\mathbf{v}_{1:t-1})$
  $\boldsymbol{\Sigma}_{hh} \leftarrow \mathbf{A}\mathbf{F}\mathbf{A}^\mathsf{T} + \boldsymbol{\Sigma}_h, \quad \boldsymbol{\Sigma}_{vv} \leftarrow \mathbf{B}\boldsymbol{\Sigma}_{hh}\mathbf{B}^\mathsf{T} + \boldsymbol{\Sigma}_v, \quad \boldsymbol{\Sigma}_{vh} \leftarrow \mathbf{B}\boldsymbol{\Sigma}_{hh}$ ▷ Covariance of $p(\mathbf{h}_t, \mathbf{v}_t|\mathbf{v}_{1:t-1})$
  $\mathbf{f}' \leftarrow \boldsymbol{\mu}_h + \boldsymbol{\Sigma}_{vh}^\mathsf{T}\boldsymbol{\Sigma}_{vv}^{-1}(\mathbf{v} - \boldsymbol{\mu}_v), \quad \mathbf{F}' \leftarrow \boldsymbol{\Sigma}_{hh} - \boldsymbol{\Sigma}_{vh}^\mathsf{T}\boldsymbol{\Sigma}_{vv}^{-1}\boldsymbol{\Sigma}_{vh}$ ▷ Find $p(\mathbf{h}_t|\mathbf{v}_{1:t})$ by conditioning
  $p' \leftarrow \exp\left(-\frac{1}{2}(\mathbf{v} - \boldsymbol{\mu}_v)^\mathsf{T}\boldsymbol{\Sigma}_{vv}^{-1}(\mathbf{v} - \boldsymbol{\mu}_v)\right) / \sqrt{\det(2\pi\boldsymbol{\Sigma}_{vv})}$ ▷ Compute $p(\mathbf{v}_t|\mathbf{v}_{1:t-1})$
  return $\mathbf{f}', \mathbf{F}', p'$
end function

Algorithm 24.1 from Barber

The filtering algorithm in this case is the famous Kalman filter

- not much more than completing the square!

The linear-Gaussian model has many applications in engineering, economics and time series analysis.

Algorithm 24.2 LDS Backward Pass. Compute the smoothed posteriors $p(\mathbf{h}_t|\mathbf{v}_{1:T})$. This requires the filtered results from algorithm(24.1).

$\mathbf{G}_T \leftarrow \mathbf{F}_T$, $\mathbf{g}_T \leftarrow \mathbf{f}_T$

for $t \leftarrow T-1, 1$ **do**

$\{\mathbf{g}_t, \mathbf{G}_t\} = \text{LDSBACKWARD}(\mathbf{g}_{t+1}, \mathbf{G}_{t+1}, \mathbf{f}_t, \mathbf{F}_t; \theta_t)$

end for

function LDSBACKWARD$(\mathbf{g}, \mathbf{G}, \mathbf{f}, \mathbf{F}; \theta)$

$\boldsymbol{\mu}_h \leftarrow \mathbf{A}\mathbf{f} + \bar{\mathbf{h}}, \quad \boldsymbol{\Sigma}_{h'h'} \leftarrow \mathbf{A}\mathbf{F}\mathbf{A}^\mathsf{T} + \boldsymbol{\Sigma}_h, \quad \boldsymbol{\Sigma}_{h'h} \leftarrow \mathbf{A}\mathbf{F}$ ▷ Statistics of $p(\mathbf{h}_t, \mathbf{h}_{t+1}|\mathbf{v}_{1:t})$

$\overleftarrow{\boldsymbol{\Sigma}} \leftarrow \mathbf{F} - \boldsymbol{\Sigma}_{h'h}^\mathsf{T}\boldsymbol{\Sigma}_{h'h'}^{-1}\boldsymbol{\Sigma}_{h'h}, \quad \overleftarrow{\mathbf{A}} \leftarrow \boldsymbol{\Sigma}_{h'h}^\mathsf{T}\boldsymbol{\Sigma}_{h'h'}^{-1}, \quad \overleftarrow{\mathbf{m}} \leftarrow \mathbf{f} - \overleftarrow{\mathbf{A}}\boldsymbol{\mu}_h$ ▷ Dynamics Reversal $p(\mathbf{h}_t|\mathbf{h}_{t+1}, \mathbf{v}_{1:t})$

$\mathbf{g}' \leftarrow \overleftarrow{\mathbf{A}}\mathbf{g} + \overleftarrow{\mathbf{m}}, \quad \mathbf{G}' \leftarrow \overleftarrow{\mathbf{A}}\mathbf{G}\overleftarrow{\mathbf{A}}^\mathsf{T} + \overleftarrow{\boldsymbol{\Sigma}}$ ▷ Backward propagation

return $\mathbf{g}', \mathbf{G}'$

end function

Algorithm 24.2 from Barber

Figure 24.7 from Barber shows an application of a linear-Gaussian model:

- $H = 6$-dimensional representing location, velocity and acceleration of object in (x, y)-coordinates.
- $V = 2$-dimensional representing noisy, i.e. Gaussian, observations of the (x, y)-coordinates of the object.
- ODEs describing Newton's laws are discretized and result in a linear system.

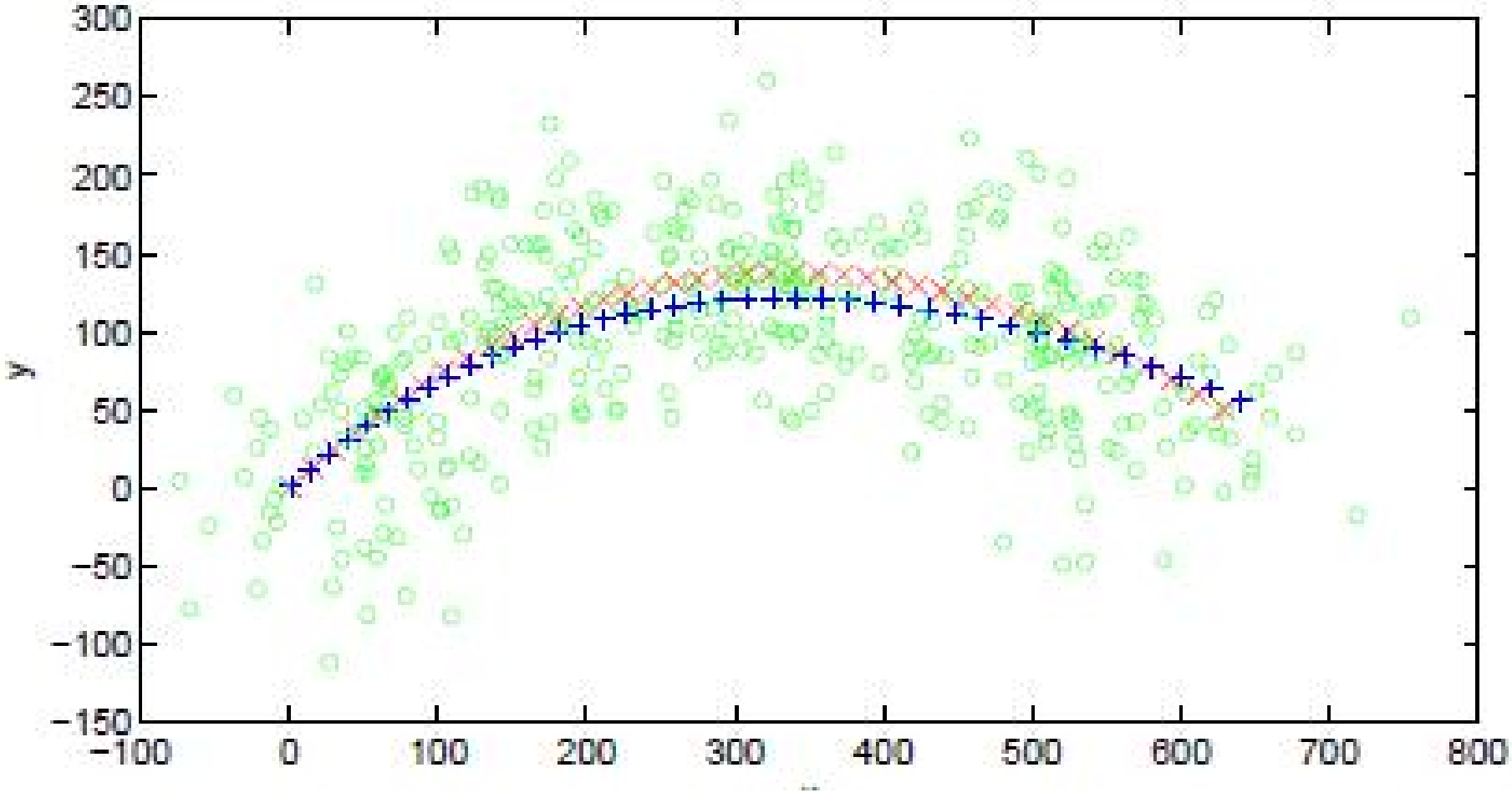


Figure 24.7 from Barber: Estimate of the trajectory of a Newtonian ballistic object based on noisy observations (small circles). All time labels are known but omitted in the plot. The 'x' points are the true positions of the object, and the crosses '+' are the estimated smoothed mean positions $\langle x_t, y_t \,|\, \mathbf{v}_{1:T} \rangle$ of the object plotted every several time steps.

Appendix: Particle Filters and Sequential Monte-Carlo

What can be done when the dynamical system is not linear-Gaussian?

Tractability then lost and we need to resort to approximate methods – and Monte-Carlo methods in particular.

Recall goal (in general) is to infer posterior distribution $\mathsf{P}(h_{1:T} \,|\, v_{1:T})$.

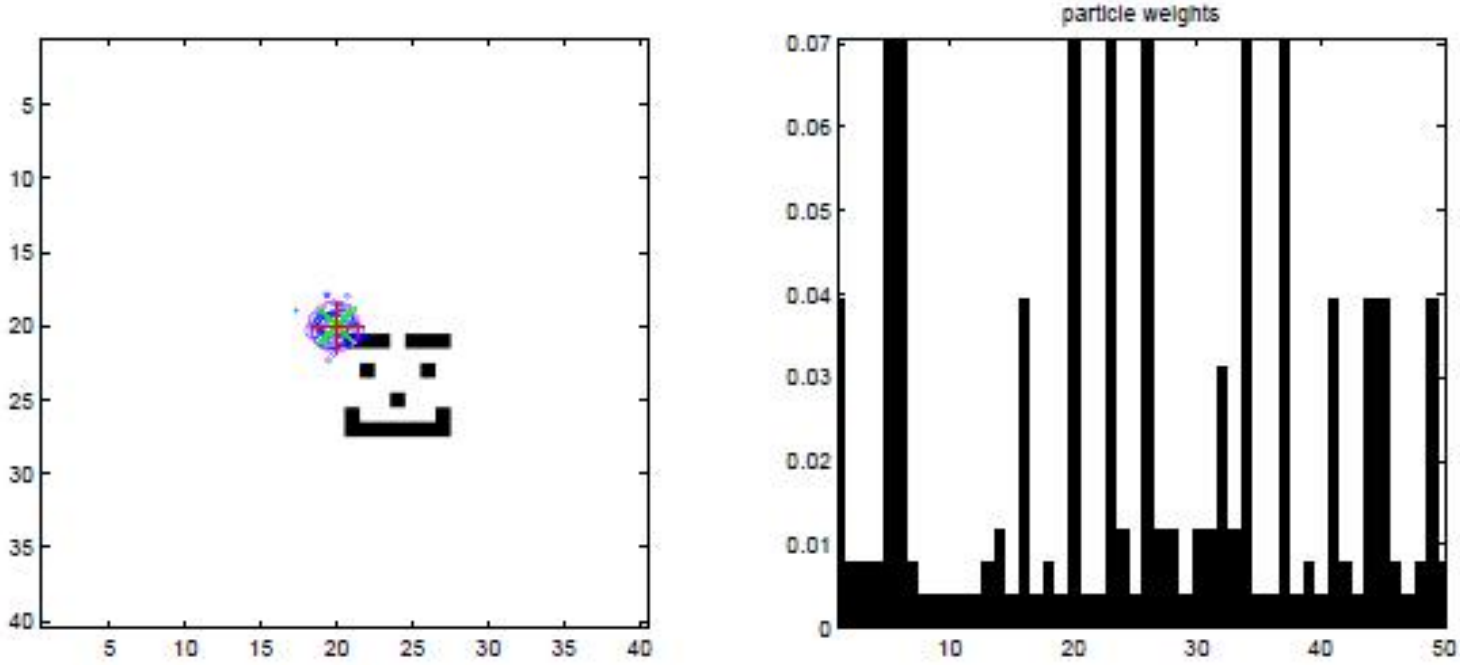
Can't sample directly but could use MCMC since we know $\mathsf{P}(h_{1:T}, v_{1:T})$

But MCMC can be very slow and need to re-run every time a new observation, v_t, appears.

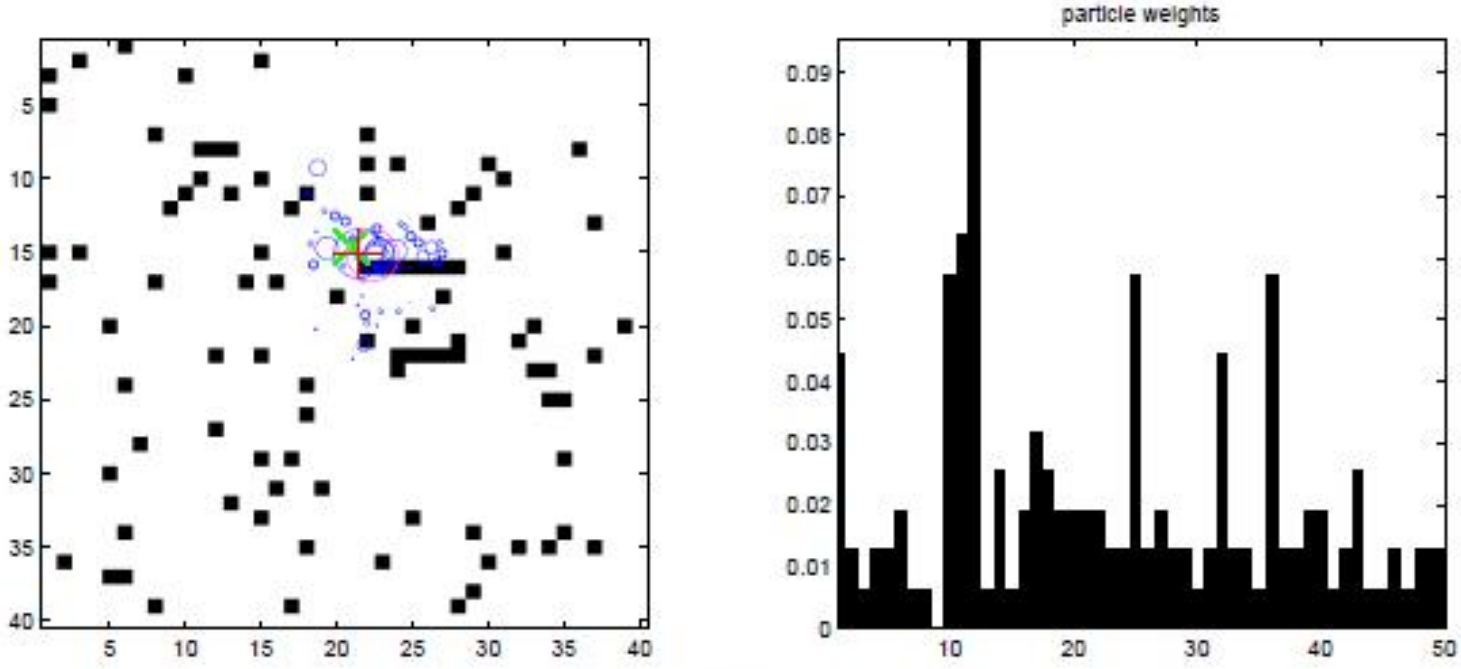
Instead can use Sequential Monte-Carlo – also known as particle filtering

- Sequential Monte-Carlo intended to mimic the Kalman filter.
- At each time t we have N particles, $h_t^1, \ldots, h_t^N$, that together with the "weights", $w_t^1, \ldots, w_t^N$, are intended to "represent" $\mathsf{P}(h_t \,|\, v_{1:t})$.

  When a new observation, v_{t+1}, arrives we sample N new particles from this distribution and then re-weight them appropriately
- Can also run some MCMC steps if so desired.



(a)



Previous slide: Figure 27.15 from Barber: Tracking an object with a particle filter containing 50 particles. The small circles are the particles, scaled by their weights. The correct corner position of the face is given by the 'x', the filtered average by the large circle 'o', and the most likely particle by '+'. (a): Initial position of the face without noise and corresponding weights of the particles. (b): Face with noisy background and the tracked corner position after 20 time-steps. The Forward-Sampling-Resampling PF method is used to maintain a healthy proportion of non-zero weights.

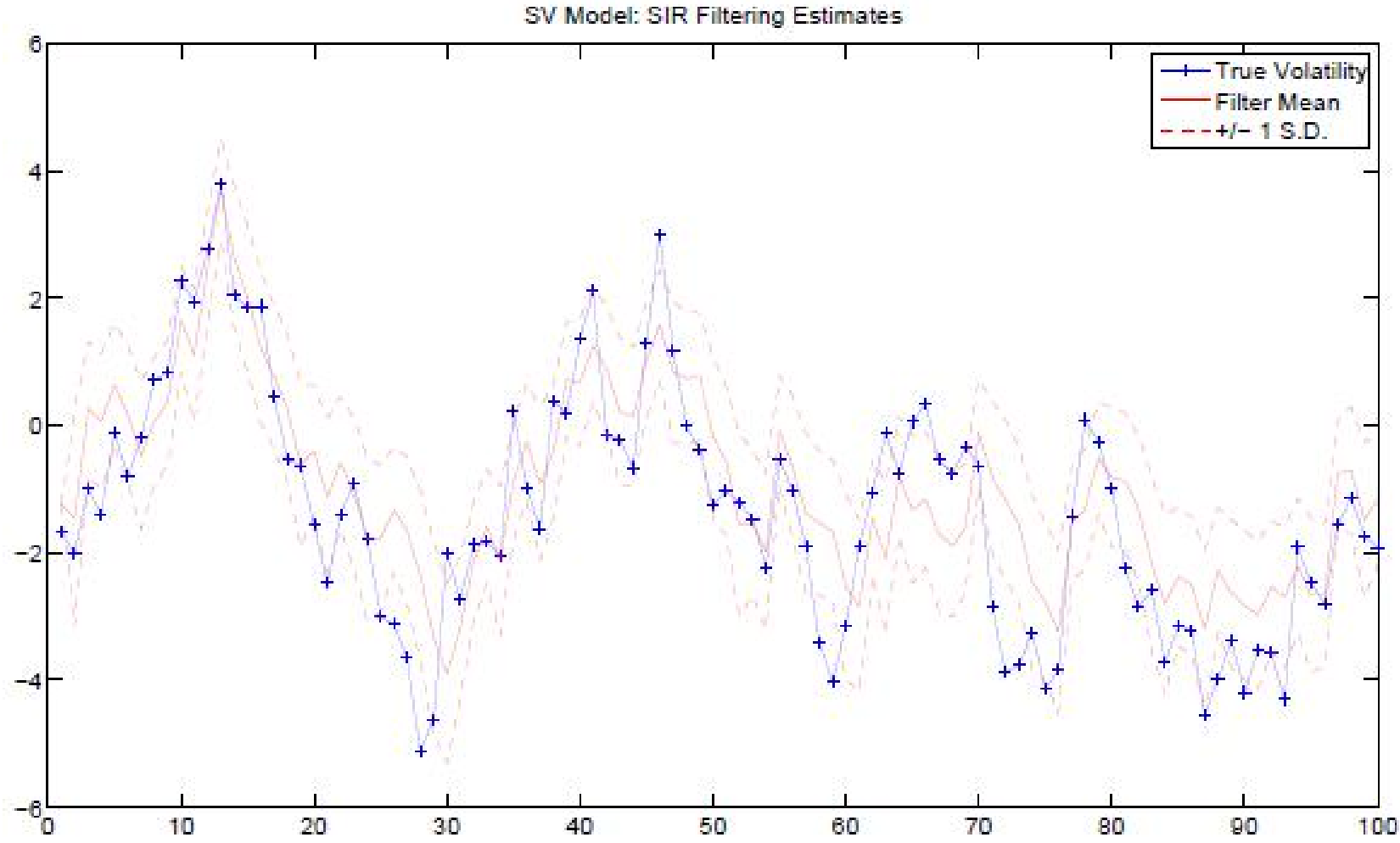


Figure 5 from Doucet and Johansen (2008): SIR filtering estimates for the SV model.

Figure 5 depicts an application to a stochastic volatility model

- the true volatility is not observed but the return data are.