DS-GA 3001.001 Special Topics in Data Science: Modeling Time Series
Handout: HMMs

1 HMM model description

Hidden Markov models are a specific instance of latent state models where variables $\mathbf{z}_i$ can take one out of K values. This is a time-series generalization of a mixture model, with the added dependency structure, capturing for instance the observation that data points that are close together in time tend to come from the same mixture component. The component densities are specified by the emission probabilities: $\mathrm{P}\left(\mathbf{x}_i|\mathbf{z}_i,\phi\right)$ (in principle this could be any distribution, need not be gaussian).

To make the later derivations easier to follow we use a one-in-k notation for the latent state, where the latent variables are K-dimensional binary vectors, with exactly one nonzero entry; $\mathbf{z}_{i,k}=1$ denotes the fact that the observation $\mathbf{x}_i$ comes from component k. The transition probabilities between latent states are specified by matrix $\mathbf{A}$:

$$A_{jk}=\mathrm{P}\left(\mathbf{z}_{i+1,k}|\mathbf{z}_{i,j}\right) \tag{1}$$

where $\sum_k A_{jk}=1$. The distribution of the initial state $\mathbf{z}_1$ is specified by parameter $\boldsymbol{\pi}$, with $\sum_k \pi_k=1$. Since entries are binary, the transition probabilities can be rewritten as a product (useful when writing out the log likelihood later):

$$\begin{aligned}\mathrm{P}\left(\mathbf{z}_{i+1}|\mathbf{z}_i\right) &= \prod_{j,k} A_{jk}^{z_{i+1,k}z_{i,j}} &(2)\\ \mathrm{P}\left(\mathbf{z}_1\right) &= \prod_k \pi_k^{z_{1,k}} &(3)\end{aligned}$$

2 Inference in HMMs

Considering all possible assignments for the latent states $\mathbf{z}_{1:t}$ involves K^t configurations (exponential in data size, intractable), nonetheless, we can efficiently compute first and (some of the) second moments of the posterior by a version of message passing.

We represent the joint posterior marginals using quantities:

$$\begin{aligned}\gamma(\mathbf{z}_i) &= \mathrm{P}\left(\mathbf{z}_i|\mathbf{x}_*,\theta^{\mathrm{old}}\right) &(4)\\ \xi(\mathbf{z}_i,\mathbf{z}_{i+1}) &= \mathrm{P}\left(\mathbf{z}_i,\mathbf{z}_{i+1}|\mathbf{x}_*,\theta^{\mathrm{old}}\right) &(5)\end{aligned}$$

Similarly, we denote the conditional expectation of the first and second moment as:[1]

$$\begin{aligned}\gamma(z_{ik}) &= \mathbb{E}[z_{i,k}|\mathbf{x}_*,\theta^{\mathrm{old}}]=\sum_{\mathbf{z}_i}\gamma(\mathbf{z}_i)z_{i,k} &(6)\\ \xi(z_{i,j},z_{i+1,k}) &= \mathbb{E}[z_{i,j}z_{i+1,k}|\mathbf{x}_*,\theta^{\mathrm{old}}]=\sum_{\mathbf{z}_i,\mathbf{z}_{i+1}}\xi(\mathbf{z}_i,\mathbf{z}_{i+1})z_{i,j}z_{i+1,k} &(7)\end{aligned}$$

For each time point γ is a K-dimensional vector, and ξ a $K\times K$ matrix.

We seek an efficient way to evaluate $\gamma(z_{i,k})$ and $\xi(z_{i,j},z_{i+1,k})$. As for LDS, inference involves propagating a pair of different messages left- and right-wards on the dependency chain, a procedure usually referred to as the forward-backward algorithm, or as the Baum-Welch algorithm. There are several variants of this basic scheme, corresponding to slightly different ways of defining the individual messages, which all yield the exact marginals; here we focus on the most common version - the *alpha-beta* algorithm (at the end we're see how Kalman filtering and smoothing are a variation of the same idea).

[1] Remember notational convention: bold for vectors, regular for individual scalar components.

Before we start, here are few useful conditional independence relations which fall from the graphical model for HMMs (also true for LDS):

$$\begin{aligned}
\mathrm{P}\left(\mathbf{x}_{1:t}|\mathbf{z}_i\right) &= \mathrm{P}\left(\mathbf{x}_{1:i}|\mathbf{z}_i\right)\mathrm{P}\left(\mathbf{x}_{i+1:t}|\mathbf{z}_i\right) && (8)\\
\mathrm{P}\left(\mathbf{x}_{1:i}|\mathbf{z}_{i+1},\mathbf{x}_{i+1}\right) &= \mathrm{P}\left(\mathbf{x}_{1:i}|\mathbf{z}_{i+1}\right) && (9)\\
\mathrm{P}\left(\mathbf{x}_{i+2:t}|\mathbf{z}_{i+1},\mathbf{x}_{i+1}\right) &= \mathrm{P}\left(\mathbf{x}_{i+2:t}|\mathbf{z}_{i+1}\right) && (10)\\
\mathrm{P}\left(\mathbf{x}_{1:i}|\mathbf{z}_i,\mathbf{z}_{i+1}\right) &= \mathrm{P}\left(\mathbf{x}_{1:i}|\mathbf{z}_i\right) && (11)\\
\mathrm{P}\left(\mathbf{x}_{i+1:t}|\mathbf{z}_i,\mathbf{z}_{i+1}\right) &= \mathrm{P}\left(\mathbf{x}_{i+1:t}|\mathbf{z}_{i+1}\right) && (12)\\
\mathrm{P}\left(\mathbf{x}_{1:t}|\mathbf{z}_i,\mathbf{z}_{i+1}\right) &= \mathrm{P}\left(\mathbf{x}_{1:i}|\mathbf{z}_i\right)\mathrm{P}\left(\mathbf{x}_{i+1}|\mathbf{z}_{i+1}\right)\mathrm{P}\left(\mathbf{x}_{i+2:t}|\mathbf{z}_{i+1}\right) && (13)
\end{aligned}$$

First, to compute $\gamma(z_{i,k})$, we need the marginal probability $\mathrm{P}\left(\mathbf{z}_i|\mathbf{x}_{1:t},\theta^{\text{old}}\right)$, which we can rewrite using Bayes' rule and expression 8 as:

$$\mathrm{P}\left(\mathbf{z}_i|\mathbf{x}_{1:t},\theta^{\text{old}}\right) = \frac{\mathrm{P}\left(\mathbf{x}_{1:t}|\mathbf{z}_i\right)\mathrm{P}\left(\mathbf{z}_i\right)}{\mathrm{P}\left(\mathbf{x}_{1:t}\right)} = \frac{\mathrm{P}\left(\mathbf{x}_{1:i},\mathbf{z}_i\right)\mathrm{P}\left(\mathbf{x}_{i+1:t}|\mathbf{z}_i\right)}{\mathrm{P}\left(\mathbf{x}_{1:t}\right)} = \frac{\alpha(\mathbf{z}_i)\beta(\mathbf{z}_i)}{\mathrm{P}\left(\mathbf{x}_{1:t}\right)} \qquad (14)$$

where we defined the two factors in the numerator as:

$$\begin{aligned}
\alpha(\mathbf{z}_i) &= \mathrm{P}\left(\mathbf{x}_{1:i},\mathbf{z}_i\right) && (15)\\
\beta(\mathbf{z}_i) &= \mathrm{P}\left(\mathbf{x}_{i+1:t}|\mathbf{z}_i\right) && (16)
\end{aligned}$$

Using the dependency structure of HMMs we can derive recursive equations for these quantities:

$$\begin{aligned}
\alpha(\mathbf{z}_i) &= \mathrm{P}\left(\mathbf{x}_{1:i},\mathbf{z}_i\right) = \mathrm{P}\left(\mathbf{x}_{1:i}|\mathbf{z}_i\right)\mathrm{P}\left(\mathbf{z}_i\right) && (17)\\
&= \mathrm{P}\left(\mathbf{x}_{1:i-1}|\mathbf{z}_i\right)\mathrm{P}\left(\mathbf{x}_i|\mathbf{z}_i\right)\mathrm{P}\left(\mathbf{z}_i\right) && (18)\\
&= \mathrm{P}\left(\mathbf{x}_i|\mathbf{z}_i\right)\sum_{\mathbf{z}_{i-1}}\mathrm{P}\left(\mathbf{x}_{1:i-1},\mathbf{z}_{i-1},\mathbf{z}_i\right) && (19)\\
&= \mathrm{P}\left(\mathbf{x}_i|\mathbf{z}_i\right)\sum_{\mathbf{z}_{i-1}}\mathrm{P}\left(\mathbf{x}_{1:i-1}|\mathbf{z}_{i-1},\mathbf{z}_i\right)\mathrm{P}\left(\mathbf{z}_i,\mathbf{z}_{i-1}\right) && (20)\\
&= \mathrm{P}\left(\mathbf{x}_i|\mathbf{z}_i\right)\sum_{\mathbf{z}_{i-1}}\mathrm{P}\left(\mathbf{x}_{1:i-1}|\mathbf{z}_{i-1}\right)\mathrm{P}\left(\mathbf{z}_i|\mathbf{z}_{i-1}\right)\mathrm{P}\left(\mathbf{z}_{i-1}\right) && (21)\\
&= \mathrm{P}\left(\mathbf{x}_i|\mathbf{z}_i\right)\sum_{\mathbf{z}_{i-1}}\mathrm{P}\left(\mathbf{x}_{1:i-1},\mathbf{z}_{i-1}\right)\mathrm{P}\left(\mathbf{z}_i|\mathbf{z}_{i-1}\right) && (22)\\
&= \mathrm{P}\left(\mathbf{x}_i|\mathbf{z}_i\right)\sum_{\mathbf{z}_{i-1}}\alpha(\mathbf{z}_{i-1})\mathrm{P}\left(\mathbf{z}_i|\mathbf{z}_{i-1}\right) && (23)
\end{aligned}$$

where the initial condition is gives as:

$$\alpha(\mathbf{z}_1) = \mathrm{P}\left(\mathbf{x}_1,\mathbf{z}_1\right) = \mathrm{P}\left(\mathbf{x}_1|\mathbf{z}_1\right)\mathrm{P}\left(\mathbf{z}_1\right) = \prod_k\left(\pi_k\mathrm{P}\left(\mathbf{x}_1|\phi_k\right)\right)^{z_{1k}}. \qquad (24)$$

Similarly, we can compute the recursion equations for β as:

$$\begin{aligned}
\beta(\mathbf{z}_i) &= \mathrm{P}\left(\mathbf{x}_{i+1:t}|\mathbf{z}_i\right) && (25)\\
&= \sum_{\mathbf{z}_{i+1}}\mathrm{P}\left(\mathbf{x}_{i+1:t},\mathbf{z}_i|\mathbf{z}_i\right) && (26)\\
&= \sum_{\mathbf{z}_{i+1}}\mathrm{P}\left(\mathbf{x}_{i+1:t}|\mathbf{z}_{i+1},\mathbf{z}_i\right)\mathrm{P}\left(\mathbf{z}_{i+1}|\mathbf{z}_i\right) && (27)\\
&= \sum_{\mathbf{z}_{i+1}}\mathrm{P}\left(\mathbf{x}_{i+1}|\mathbf{z}_{i+1}\right)\mathrm{P}\left(\mathbf{x}_{i+2:t}|\mathbf{z}_{i+1}\right)\mathrm{P}\left(\mathbf{z}_{i+1}|\mathbf{z}_i\right) && (28)\\
&= \sum_{\mathbf{z}_{i+1}}\beta(\mathbf{z}_{i+1})\mathrm{P}\left(\mathbf{x}_{i+1}|\mathbf{z}_{i+1}\right)\mathrm{P}\left(\mathbf{z}_{i+1}|\mathbf{z}_i\right) && (29)\\
& && (30)
\end{aligned}$$

This is a backward message that determines value at time i as a function of value at time $i+1$. The starting point for the propagation of this signal is $\beta(\mathbf{z}_t) = 1$, obtained from the definition of the posterior at the extreme:

$$\mathrm{P}\left(\mathbf{z}_t|\mathbf{x}_{1:t}\right) = \frac{\mathrm{P}\left(\mathbf{x}_{1:t}, \mathbf{z}_t\right)\beta(\mathbf{z}_t)}{\mathrm{P}\left(\mathbf{x}_{1:t}\right)}$$

Lastly, the likelihood $\mathrm{P}\left(\mathbf{x}_{1:t}\right)$ can be computed at any point as the normalizing constant $\mathrm{P}\left(\mathbf{x}_{1:t}\right) = \sum_{\mathbf{z}_k} \alpha(\mathbf{z}_k)\beta(\mathbf{z}_k)$, or using $\beta(\mathbf{z}_t) = 1$, as $\mathrm{P}\left(\mathbf{x}_{1:t}\right) = \sum_{\mathbf{z}_t} \alpha(\mathbf{z}_t)$.

We now need to compute the joint statistics $\xi(z_{i,j}, z_{i+1,k})$; using Bayes rule, factorization 13, and the definitions for α and β this quantity becomes:

$$\begin{aligned}
\mathrm{P}\left(\mathbf{z}_i, \mathbf{z}_{i+1}|\mathbf{x}_{1:t}\right) &= \frac{\mathrm{P}\left(\mathbf{x}_{1:t}|\mathbf{z}_i, \mathbf{z}_{i+1}\right)\mathrm{P}\left(\mathbf{z}_i, \mathbf{z}_{i+1}\right)}{\mathrm{P}\left(\mathbf{x}_{1:t}\right)} && (31)\\
&= \frac{\mathrm{P}\left(\mathbf{x}_{1:i}|\mathbf{z}_i\right)\mathrm{P}\left(\mathbf{z}_i\right)\mathrm{P}\left(\mathbf{z}_{i+1}|\mathbf{z}_i\right)\mathrm{P}\left(\mathbf{x}_{i+1:t}|\mathbf{z}_{i+1}\right)}{\mathrm{P}\left(\mathbf{x}_{1:t}\right)} && (32)\\
&= \frac{\alpha(\mathbf{z}_i)\mathrm{P}\left(\mathbf{z}_{i+1}|\mathbf{z}_i\right)\mathrm{P}\left(\mathbf{x}_{i+1}|\mathbf{z}_{i+1}\right)\beta(\mathbf{z}_{i+1})}{\mathrm{P}\left(\mathbf{x}_{1:t}\right)} && (33)
\end{aligned}$$

so it can be computed directly using the α and β recursions.

While the above expressions are mathematically accurate, since we are representing actual probability of single events in a potentially large space (very small numbers) which means that we need to worry about numerical stability when implementing this algorithm in practice. For this purpose, we define scaled versions of α and β so that their values stay in sensible limits:

$$\begin{aligned}
\hat{\alpha}(\mathbf{z}_i) &= \mathrm{P}\left(\mathbf{z}_i|\mathbf{x}_{1:i}\right) = \frac{\alpha(\mathbf{z}_i)}{\mathrm{P}\left(\mathbf{x}_{1:i}\right)} && (34)\\
\hat{\beta}(\mathbf{z}_i) &= \frac{\mathrm{P}\left(\mathbf{x}_{i+1:t}|\mathbf{z}_i\right)}{\mathrm{P}\left(\mathbf{x}_{\mathbf{i+1:t}}|\mathbf{x}_{1:i}\right)} = \frac{\beta(\mathbf{z}_i)}{\prod_{j=i+1:t} c_j} && (35)
\end{aligned}$$

where the scaling factors are defined using an intermediate quantity, c_i:

$$\begin{aligned}
c_i &= \mathrm{P}\left(\mathbf{x}_i|\mathbf{x}_{1:i-1}\right) && (36)\\
\mathrm{P}\left(\mathbf{x}_{1:i}\right) &= \prod_{j=1:i} c_j && (37)
\end{aligned}$$

The updated recursion equation for $\hat{\alpha}$, and $\hat{\beta}$ become:

$$\begin{aligned}
c_i\hat{\alpha}(\mathbf{z}_i) &= \mathrm{P}\left(\mathbf{x}_i|\mathbf{z}_i\right)\sum_{\mathbf{z}_{i-1}} \hat{\alpha}(\mathbf{z}_{i-1})\mathrm{P}\left(\mathbf{z}_i|\mathbf{z}_{i-1}\right) && (38)\\
c_{i+1}\hat{\beta}(\mathbf{z}_i) &= \sum_{\mathbf{z}_{i+1}} \hat{\beta}(\mathbf{z}_{i+1})\mathrm{P}\left(\mathbf{x}_{i+1}|\mathbf{z}_{i+1}\right)\mathrm{P}\left(\mathbf{z}_{i+1}|\mathbf{z}_i\right) && (39)
\end{aligned}$$

Finally the new expressions for the posterior marginals are:

$$\begin{aligned}
\gamma(\mathbf{z}_i) &= \hat{\alpha}(\mathbf{z}_i)\hat{\beta}(\mathbf{z}_i) && (40)\\
\xi(z_{i,j}, z_{i+1,k}) &= {c_{i+1}}^{-1}\hat{\alpha}(\mathbf{z}_i)\mathrm{P}\left(\mathbf{z}_{i+1}|\mathbf{z}_i\right)\mathrm{P}\left(\mathbf{x}_{i+1}|\mathbf{z}_{i+1}\right)\hat{\beta}(\mathbf{z}_{i+1}) && (41)
\end{aligned}$$

Note: LDS inference can be viewed as a special case of the above recipe where the marginalization is done analytically; the forward message (Kalman filter) is exactly the corresponding $\hat{\alpha}$, while the backward message (Kalman smoother) is $\gamma = \hat{\alpha}\hat{\beta}$. Because of this, the linear dynamical systems version of the algorithm is sometimes referred to as the alpha-gamma.

3 HMMs parameter learning: EM

We us *expectation maximization* to provide maximum likelihood estimates for the model parameter. To compute the ML estimates, we need to find the parameters that maximize the likelihood function (marginalizing out the unknown latents):

$$\mathrm{P}\left(\mathbf{x}|\theta\right) = \int_Z \mathrm{P}\left(\mathbf{x}, \mathbf{z}|\theta\right) \mathbf{dz} \tag{42}$$

where $\mathbf{x}$ denotes all observed variables and $\mathbf{z}$ all latents.

Using auxiliary distribution $q(\mathbf{z})$, we can rewrite the likelihood in a more convenient form:

$$\begin{aligned}
\log \mathrm{P}\left(\mathbf{x}|\theta\right) &= \log \mathrm{P}\left(\mathbf{x}|\theta\right) \int_Z q(\mathbf{z}) \mathbf{dz} && (43)\\
&= \int_Z q(\mathbf{z}) \log \frac{\mathrm{P}\left(\mathbf{x}, \mathbf{z}|\theta\right)}{q(\mathbf{z})} \mathbf{dz} - \int_Z q(\mathbf{z}) \log \frac{\mathrm{P}\left(\mathbf{z}|\mathbf{x}, \theta\right)}{q(\mathbf{z})} \mathbf{dz} && (44)\\
&= \mathcal{L}(q, \theta) + \mathrm{KL}\left(q(\mathbf{z})||\mathrm{P}\left(\mathbf{z}|\mathbf{x}, \theta\right)\right) && (45)
\end{aligned}$$

where KL denotes the Kullback-Leibler divergence. Since $\mathrm{KL}(p||q) \geq 0$ for any pairs of distributions p, q, the functional $\mathcal{L}(q, \theta)$ is a lower bound for $\log \mathrm{P}\left(\mathbf{x}|\theta\right)$. Moreover, the bound becomes tight when the KL is zero, i.e. when $q(\mathbf{z}) = \mathrm{P}\left(\mathbf{z}|\mathbf{x}, \theta^{\mathrm{old}}\right)$.

The optimization procedure proceeds in 2 steps: the E-step optimizes $\mathcal{L}(q, \theta)$ with respect to q for fixed parameters θ^{old}, which yields $q(\mathbf{z}) = \mathrm{P}\left(\mathbf{z}|\mathbf{x}, \theta^{\mathrm{old}}\right)$. The M-step, optimizes the expectation under q of the complete data (log)likelihood $\mathcal{Q}(\theta, \theta^{\mathrm{old}}) = \int_Z \mathrm{P}\left(\mathbf{z}|\mathbf{x}, \theta^{\mathrm{old}}\right) \log \mathrm{P}\left(\mathbf{x}, \mathbf{z}|\theta\right)$ (see LDS handout for a more detailed explanation of the general steps). For HMMs the parameters to optimise are $\theta = \{\mathbf{A}, \boldsymbol{\pi}, \phi\}$

Using the above notation, we can rewrite the complete data likelihood as:

$$\begin{aligned}
\mathcal{Q}(\theta, \theta^{\mathrm{old}}) &= \sum_k \gamma(z_{0,k}) + \sum_{i,j,k} \xi(z_{i,j}, z_{i+1,k}) \log A_{jk} && (46)\\
&+ \sum_{i,k} \gamma(z_{i,k}) \log \mathrm{P}\left(\mathbf{x}_i|\phi_k\right) && (47)
\end{aligned}$$

where we have used eqs. 2 and 3 and the definition of the expectations under q, eqs. 6 and 7.

The goal of E- step is to provide a way to efficiently evaluate quantities $\gamma(z_{i,k})$ and $\xi(\mathbf{z}_i, \mathbf{z}_{i+1})$. Once those are fixed, the M-step updates the parameters:

$$\begin{aligned}
\pi_k^{\mathrm{new}} &= \frac{\gamma(z_{1,k})}{\sum_j \gamma(z_{1,j})} && (48)\\
A_{jk}^{\mathrm{new}} &= \frac{\sum_i \xi(z_{i,j}, z_{i+1,k})}{\sum_{i,l} \xi(z_{i,j}, z_{i+1,l})} && (49)
\end{aligned}$$

Essentially, we count up fuzzy class assignments, then normalize result to get probabilities.

The exact parameter update will depend on the details of the emission model, but the general idea that we use $\gamma(z_{i,k})$ as class responsibilities remains the same. As an illustration, let's consider the case where the individual components are gaussian with class specific means μ_k and covariances Σ_k:

$$\begin{aligned}
\mu_k^{\mathrm{new}} &= \frac{1}{\sum_i \gamma(z_{i,k})} \sum_i \gamma(z_{i,k}) \mathbf{x}_i && (50)\\
\Sigma_k^{\mathrm{new}} &= \frac{1}{\sum_i \gamma(z_{i,k})} \sum_i \gamma(z_{i,k}) \mathbf{x}_i \mathbf{x}_i^{\mathrm{t}} - \mu_k \mu_k^{\mathrm{t}} && (51)
\end{aligned}$$

For this observation model, we are simply computing a weighted version of the empirical mean and covariance.

4 The Viterbi algorithm

We have already discussed the fact that the posterior distribution for $\mathbf{z}_{1:t}$ is an intractable quantity, and we have seen that it is still possible to estimate its marginals efficiently. There are many applications where it is also useful to know the most likely configuration for $\mathbf{z}_{1:t}$, $\hat{\mathbf{z}}_{1:t} = \text{argmax}_{\mathbf{z}_{1:t}}\{\mathrm{P}\left(\mathbf{z}_{1:t}|\mathbf{x}_{1:t}\right)\}$, or the location of the peak of the posterior distribution (note that in general this not the same as the peak of the marginals).

Finding the peak of the posterior (sometimes referred to as maximum a posteriori, or MAP, inference) for HMMs is a special instance of the max-sum algorithm (but we do not discuss this link in detail here, see Bishop if interested). Alternatively, it can be intuitively understood as a dynamic programming procedure, where if we are given the probability of the most likely configuration for state $\mathbf{z}_{1:i}|\mathbf{z}_{1:i}$ we can combine this with the observation $\mathbf{x}_i$ to determine the most likely configuration $\mathbf{z}_{1:i+1}|\mathbf{z}_{1:i+1}$ and it's associated probability. In practice, this looks very similar to the forward message passing for an HMM (or the kalman filtering), with the important distinction than instead of summing over possible configurations leading to $\mathbf{z}_{i+1}$, instead we are taking the max, thus considering only the most likely path. To do this, we define a new forward message, w_i:

$$w_i = \text{max}_{\mathbf{z}_{1:i-1}} \log \mathrm{P}\left(\mathbf{z}_{1:i}, \mathbf{x}_{1:i}\right) \tag{52}$$

where we move the representation in log space to avoid the kind of numerical issues discussed when rescaling the α, β messages above. The recursive update equations for the new message are (just as before, with max instead of sum):

$$w_{i+1} = \log \mathrm{P}\left(\mathbf{x}_{i+1}|\mathbf{z}_{i+1}\right) + \text{max}_{\mathbf{z}_i}\{\log \mathrm{P}\left(\mathbf{z}_{i+1}|\mathbf{z}_i\right) + w_i(\mathbf{z}_i)\} \tag{53}$$

with the initial conditions $w_1 = \log \mathrm{P}\left(\mathbf{z}_1\right) + \log \mathrm{P}\left(\mathbf{x}_1|\mathbf{z}_1\right)$. It is easy to see that at the end of the forward run, $w_t = \text{max}_{\mathbf{z}_{1:t-1}} \log \mathrm{P}\left(\mathbf{z}_{1:t}, \mathbf{x}_{1:t}\right)$ gives the log probability of the MAP solution, but of course we don't actually just need the value of that probability but the actual sequence that generated it. For this reason, we need to additionally keep track at each step of a backward pointer memorizing which was the optimal intermediate path that got us there $k_i^{\max} = \text{argmax}_{\mathbf{z}_i}\{\log \mathrm{P}\left(\mathbf{z}_{i+1}|\mathbf{z}_i\right) + w_i(\mathbf{z}_i)\}$. A more detailed explanation for the implementation of this idea in pseudocode is provided in section 13.2.5 in Bishop.