Computing marginals, version two

We can apply sum-product to our E step graph.

\[
\begin{align*}
    z_1 & \quad z_2 & \ldots & \quad z_{n-1} & \quad z_n & \quad z_{n+1} \\
    x_1 & \quad x_2 & \quad \ldots & \quad x_{n-1} & \quad x_n & \quad x_{n+1}
\end{align*}
\]

Factor graph

\[
\begin{align*}
    \chi & \quad z_1 & \quad \ldots & \quad z_{n-1} & \quad \psi_n & \quad z_n \\
    \quad g_1 & \quad \ldots & \quad g_{n-1} & \quad g_n
\end{align*}
\]

Simplified factor graph

\[
\begin{align*}
    \chi & \quad x_1 & \quad \ldots & \quad x_{n-1} & \quad x_n & \quad \ldots \\
    \quad z_1 & \quad \ldots & \quad z_{n-1} & \quad z_n & \quad \ldots
\end{align*}
\]

Review of sum-product concepts

\[
\begin{align*}
    h & \quad z_1 & \quad \ldots & \quad z_{n-1} & \quad f_n & \quad z_n \\
    \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots & \quad \ldots
\end{align*}
\]

The marginal for each node is a product of the incoming messages.

This is analogous to setting up the marginal as a product of alpha and beta factors in the previous treatment.

Since we have a chain, this is just two messages, one coming from the left, the other from the right.

To compute all marginals, we pass the left and right messages from one end to the other.
Sum-product for HMM

\[ h = p(z_1)p(x_1|z_1) \]

extra for the nodes we pruned

Factor node actions on left to right messages

\[ \mu_{f_n \rightarrow f_{n+1}}(z_n) = \sum_{z_{n-1}} f_n(z_{n-1}, z_n) \mu_{f_{n-1} \rightarrow f_n}(z_{n-1}) \]

The first message is

\[ h = p(z_1)p(x_1|z_1) = p(x_1, z_1) = \alpha(z_1) \]
Sum-product for HMM

If we identify \( \mu_{f_n\rightarrow f_{n+1}}(z_n) = \alpha(z_n) \)

\[
\alpha(z_n) = \sum_{z_{n-1}} f_n(z_n, z_{n-1}) \alpha(z_{n-1}) \\
= \sum_{z_{n-1}} p(z_n|z_{n-1}) p(x_n|z_n) \alpha(z_{n-1}) \\
= p(x_n|z_n) \sum_{z_{n-1}} p(z_n|z_{n-1}) \alpha(z_{n-1})
\]

Identify \( \beta(z_n) \equiv \mu_{f_{n+1}\rightarrow f_n}(z_n) \) to get

\[
\beta(z_n) = \sum_{z_{n+1}} f_{n+1}(z_n, z_{n+1}) \beta(z_{n+1})
\]

Recalling that \( f_{n+1} = p(z_{n+1}|z_n) p(x_{n+1}|z_{n+1}) \)

\[
\beta(z_n) = \sum_{z_{n+1}} p(z_{n+1}|z_n) p(x_{n+1}|z_{n+1}) \beta(z_{n+1})
\]
Sum-product for E step in the HMM learning problem (review)

Given all $\alpha(z_n)$ and $\beta(z_n)$

$$\gamma(z_n) = \frac{\alpha(z_n) \beta(z_n)}{p(X)}$$

$$p(X) = \sum_{z_n} \alpha(z_n) \beta(z_n)$$

$$\hat{\xi}(z_{n-1}, z_n) = \frac{\alpha(z_{n-1}) p(x_n | z_n) p(z_n | z_{n-1}) \beta(z_n)}{p(X)}$$

Rescaled alpha beta (Bishop, 13.2.4)

The alpha-beta algorithm has similar precision problems to the ones for EM where we discussed the fix of scaling log quantities by the max, before exponentiation for normalizing.

One way to handle this is to reformulate the alpha-beta algorithm in terms of:

$$\hat{\alpha}(z_n) = p(z_n | x_1, ..., x_n) = \frac{\alpha(z_n)}{p(x_1, ..., x_n)}$$

$$\hat{\beta}(z_n) = \frac{p(x_{n+1}, ..., x_N | z_n)}{p(x_{n+1}, ..., x_N | x_1, ..., x_n)}$$

Classic HMM computational problems

Given data, what is the HMM (learning). ✓

Given an HMM, what is the distribution over the state variables. Also, how likely are the observations, given the model. ✓

Given an HMM, what is the most likely state sequence for some data?

Viterbi algorithm (special case of max-sum)

Recall max-sum

Forward direction is like sum-product, except
- We take the max instead of sum
- We use sum of logs instead of product
- We remember incoming variable values that give max (*)

Backwards direction is simply backtracking on (*).
Recall simplified factor graph

\[ h = p(z_1)p(x_1|z_1) \quad f_n = p(z_n|z_{n-1})p(x_n|z_n) \]

Left to right messages

\[ \omega(z_n) = \log(x_n|z_n) + \max_{z_{n-1}} \{ \log(p(z_n|z_{n-1}) + \omega(z_{n-1})) \} \]

\[ \omega(z_1) = \log(p(z_1)) + \log(p(x_1|z_1)) \]

Intuitive understanding

The message encodes the probabilities for the maximum probability path for each of the \( K \) states.

For each state \( k \)

Consider getting there from each previous state \( k' \)

The message is the vector of probabilities for the maximum probability path for each of the \( K \) states.

\[ \omega(z_n) = \log(x_n|z_n) + \max_{z_{n-1}} \{ \log(p(z_n|z_{n-1}) + \omega(z_{n-1})) \} \]

Consider getting there from each previous state \( k' \)

For each state \( k \)

We can see that this is the new maximum

For Viterbi, we need to remember the previous state, \( k' \), for each \( k \).
Intuitive understanding

The max path is shown (but we only know it when we get to the end).

To find the path, we need to chase the back pointers.

Final comments on learning

In many applications, the states have specified meaning, and are available in training data, so EM is not needed.

(Most authors still call this an HMM because states are hidden when the model is used).

We described training the HMM based on a single data sequence, but often multiple sequences that come from the same HMM are used (modifying the algorithm is very straightforward).

Two HMM examples (specified states)

Domain is SLIC (Semantically Linked Instructional Content).

1) Temporal information for matching video frames to slides.

2) Aligning noisy speech transcripts with slides.

Matching slides to video frames
Matching slides to video frames

\[
p(X, Z | \theta) = p(z_1 | \pi) \left[ \prod_{n=2}^{N} p(z_n | z_{n-1}, A) \right] \prod_{m=1}^{N} p(x_m | z_m, \phi)
\]

Our state sequence corresponds to what slide is being shown.

Matching slides to video frames

\[
p(X, Z | \theta) = p(z_1 | \pi) \left[ \prod_{n=2}^{N} p(z_n | z_{n-1}, A) \right] \prod_{m=1}^{N} p(x_m | z_m, \phi)
\]

From image matching

Matching slides to video frames

\[
p(z_n | z_{n-1}, A) = f(z_n - z_{n-1}) \quad \text{encodes slide jump statistics.}
\]

We assume that only the jump matters. IE, going from slide 6 to 8 has the same chance of going from 10 to 12.

\[
p(z_i | \pi) \left[ \prod_{n=2}^{N} p(z_n | z_{n-1}, A) \right]
\]

says how likely a sequence is, without looking at the images.

Matching slides to video frames

\[
p(X, Z | \theta) = p(z_1 | \pi) \left[ \prod_{n=2}^{N} p(z_n | z_{n-1}, A) \right] \prod_{m=1}^{N} p(x_m | z_m, \phi)
\]

Why bother?

Mistakes in speech transcripts can be corrected. Speech transcripts are noisy and to poorly on jargon. But jargon words often appear on slides.

We can highlight or auto-laser-point what the speaker is pointing to.

We can improve close-captioning.
Aligning speech to slides

A reasonable model for some speakers is that they say some approximation of their bullet points, with some extra stuff before and after.

Automated speech recognizers try to produce results that are plausible on a phoneme level.

If a slide word is used, its phoneme sequence will likely be approximated in the phonemes in the speech transcript.

We can calibrate the phoneme “confusion matrix.”

**Aligned speech for correction**

Speaker says: maliciousness
ASR produces: my dishes nests  
Slide word: maliciousness  
with slide word phoneme sequence

If the same mistake is made later, where the word is not on the slide, we can propagate the correction.