

HMM Algorithms

Peter Bell

Automatic Speech Recognition— ASR Lecture 5
25 January 2021

HMM algorithms

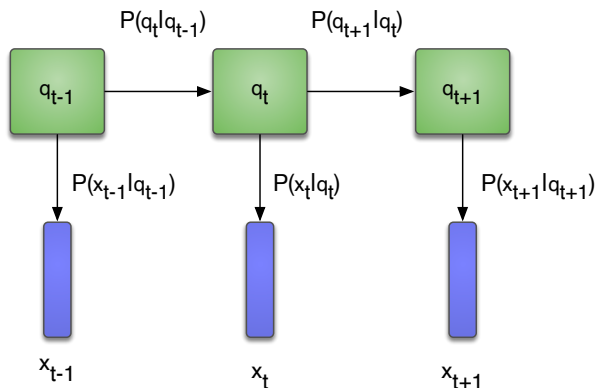
- HMM recap
- HMM algorithms (2)
 - Likelihood computation (forward algorithm)
 - Finding the most probable state sequence (Viterbi algorithm)
 - Estimating the parameters (forward-backward and EM algorithms)

HMM algorithms

- HMM recap
- HMM algorithms (2)
 - Likelihood computation (forward algorithm)
 - Finding the most probable state sequence (Viterbi algorithm)
 - Estimating the parameters (forward-backward and EM algorithms)

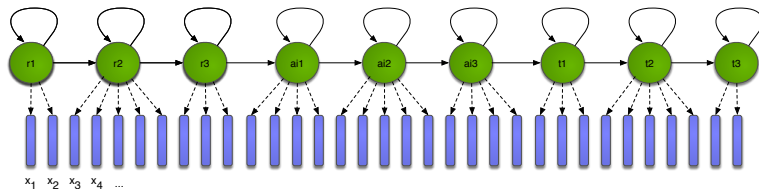
Warning: the maths continues!

Recap: the HMM



- A *generative model* for the sequence $X = (x_1, \dots, x_T)$
- Discrete states q_t are unobserved
- q_{t+1} is conditionally independent of q_1, \dots, q_{t-1} , given q_t
- Observations x_t are conditionally independent of each other, given q_t .

The three-state left-to-right topology for phones:



Computing likelihoods with the HMM

Joint likelihood of X and $Q = (q_1, \dots, q_T)$:

$$P(X, Q|\lambda) = P(q_1)P(x_1|q_1)P(q_2|q_1)P(x_2|q_2)\dots \quad (1)$$

$$= P(q_1)P(x_1|q_1) \prod_{t=2}^T P(q_t|q_{t-1})P(x_t|q_t) \quad (2)$$

$P(q_t)$ denotes the initial occupancy probability of each state

The parameters of the model, λ , are given by:

- Transition probabilities $a_{kj} = P(q_{t+1} = j | q_t = k)$
- Observation probabilities $b_j(x) = P(x | q = j)$

The three problems of HMMs

Working with HMMs requires the solution of three problems:

- ① **Likelihood** Determine the overall likelihood of an observation sequence $X = (x_1, \dots, x_t, \dots, x_T)$ being generated by a known HMM topology, \mathcal{M} .
→ the *forward algorithm*

The three problems of HMMs

Working with HMMs requires the solution of three problems:

- ① **Likelihood** Determine the overall likelihood of an observation sequence $X = (x_1, \dots, x_t, \dots, x_T)$ being generated by a known HMM topology, \mathcal{M} .
→ the *forward algorithm*
- ② **Decoding and alignment** Given an observation sequence and an HMM, determine the most probable hidden state sequence
→ the *Viterbi algorithm*

The three problems of HMMs

Working with HMMs requires the solution of three problems:

- 1 **Likelihood** Determine the overall likelihood of an observation sequence $X = (x_1, \dots, x_t, \dots, x_T)$ being generated by a known HMM topology, \mathcal{M} .
→ the *forward algorithm*
- 2 **Decoding and alignment** Given an observation sequence and an HMM, determine the most probable hidden state sequence
→ the *Viterbi algorithm*
- 3 **Training** Given an observation sequence and an HMM, find the state occupation probabilities, in order to find the best HMM parameters λ
→ the *forward-backward* and *EM* algorithms

Viterbi algorithm

- Instead of finding the likelihood over all possible state sequences, as we do in the Forward algorithm, just consider just the most probable path:

$$P^*(X|\mathcal{M}) = \max P(X, Q|\mathcal{M})$$

Viterbi algorithm

- Instead of finding the likelihood over all possible state sequences, as we do in the Forward algorithm, just consider just the most probable path:

$$P^*(X|\mathcal{M}) = \max P(X, Q|\mathcal{M})$$

- Define likelihood of the most probable partial path in state j at time t , $V_j(t)$

Viterbi algorithm

- Instead of finding the likelihood over all possible state sequences, as we do in the Forward algorithm, just consider just the most probable path:

$$P^*(X|\mathcal{M}) = \max P(X, Q|\mathcal{M})$$

- Define likelihood of the most probable partial path in state j at time t , $V_j(t)$
- If we are performing decoding or forced alignment, then only the most likely path is needed

Viterbi algorithm

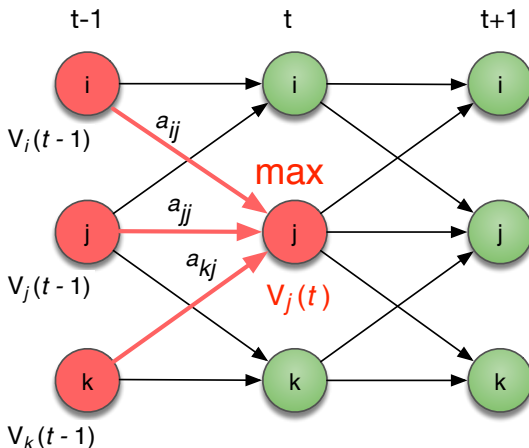
- Instead of finding the likelihood over all possible state sequences, as we do in the Forward algorithm, just consider just the most probable path:

$$P^*(X|\mathcal{M}) = \max P(X, Q|\mathcal{M})$$

- Define likelihood of the most probable partial path in state j at time t , $V_j(t)$
- If we are performing decoding or forced alignment, then only the most likely path is needed
- We need to keep track of the states that make up this path by keeping a sequence of *backpointers* to enable a Viterbi *backtrace*: the backpointer for each state at each time indicates the previous state on the most probable path

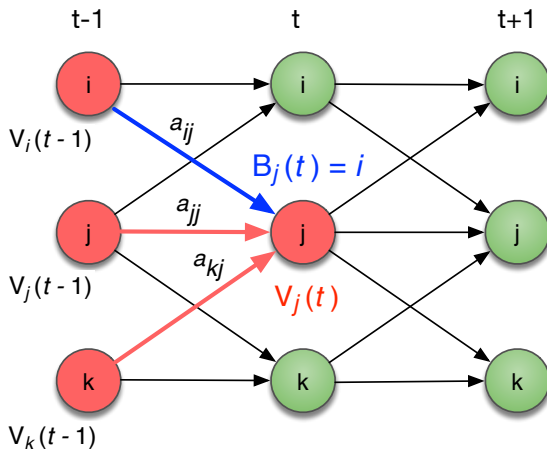
Viterbi Recursion

$$V_j(t) = \max_i V_i(t-1) a_{ij} b_j(x_t)$$



Viterbi Recursion

Backpointers to the previous state on the most probable path



2. Decoding: The Viterbi algorithm

- Initialisation

$$V_0(0) = 1$$

$$V_j(0) = 0 \quad \text{if } j \neq 0$$

$$B_j(0) = 0$$

2. Decoding: The Viterbi algorithm

- Initialisation

$$V_0(0) = 1$$

$$V_j(0) = 0 \quad \text{if } j \neq 0$$

$$B_j(0) = 0$$

- Recursion

$$V_j(t) = \max_{i=1}^J V_i(t-1) a_{ij} b_j(x_t)$$

$$B_j(t) = \arg \max_{i=1}^J V_i(t-1) a_{ij} b_j(x_t)$$

2. Decoding: The Viterbi algorithm

- Initialisation

$$V_0(0) = 1$$

$$V_j(0) = 0 \quad \text{if } j \neq 0$$

$$B_j(0) = 0$$

- Recursion

$$V_j(t) = \max_{i=1}^J V_i(t-1) a_{ij} b_j(x_t)$$

$$B_j(t) = \arg \max_{i=1}^J V_i(t-1) a_{ij} b_j(x_t)$$

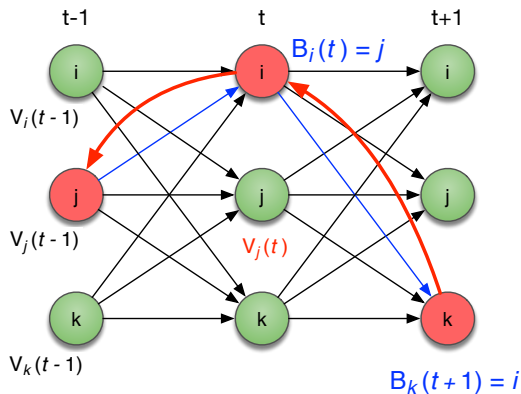
- Termination

$$P^* = V_E = \max_{i=1}^J V_T(i) a_{iE}$$

$$s_T^* = B_E = \arg \max_{i=1}^J V_i(T) a_{iE}$$

Viterbi Backtrace

Backtrace to find the state sequence of the most probable path



3. Training: Forward-Backward algorithm

- Goal: Efficiently estimate the parameters of an HMM λ from an observation sequence X and known HMM topology \mathcal{M} :

3. Training: Forward-Backward algorithm

- Goal: Efficiently estimate the parameters of an HMM λ from an observation sequence X and known HMM topology \mathcal{M} :
- Parameters λ :
 - Transition probabilities $a_{kj} = P(q_{t+1} = j | q_t = k)$
 - Observation probabilities $b_j(x) = P(x | q = j)$

3. Training: Forward-Backward algorithm

- Goal: Efficiently estimate the parameters of an HMM λ from an observation sequence X and known HMM topology \mathcal{M} :
- Parameters λ :
 - Transition probabilities $a_{kj} = P(q_{t+1} = j | q_t = k)$
 - Observation probabilities $b_j(x) = P(x | q = j)$
- Maximum likelihood training: find the parameters that maximise

$$\begin{aligned} F_{\text{ML}}(\lambda) &= \log P(X | \mathcal{M}, \lambda) \\ &= \log \sum_{Q \in \mathcal{Q}} P(X, Q | \mathcal{M}, \lambda) \end{aligned}$$

Viterbi Training

- If we knew the state-time alignment, then each observation feature vector could be assigned to a specific state

Viterbi Training

- If we knew the state-time alignment, then each observation feature vector could be assigned to a specific state
- A state-time alignment can be obtained using the most probable path obtained by Viterbi decoding

Viterbi Training

- If we knew the state-time alignment, then each observation feature vector could be assigned to a specific state
- A state-time alignment can be obtained using the most probable path obtained by Viterbi decoding
- Maximum likelihood estimate of a_{ij} , if $C(i \rightarrow j)$ is the count of transitions from i to j

$$\hat{a}_{ij} = \frac{C(i \rightarrow j)}{\sum_k C(i \rightarrow k)}$$

Viterbi Training

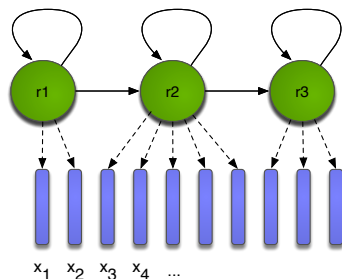
- If we knew the state-time alignment, then each observation feature vector could be assigned to a specific state
- A state-time alignment can be obtained using the most probable path obtained by Viterbi decoding
- Maximum likelihood estimate of a_{ij} , if $C(i \rightarrow j)$ is the count of transitions from i to j

$$\hat{a}_{ij} = \frac{C(i \rightarrow j)}{\sum_k C(i \rightarrow k)}$$

- Likewise if Z_j is the set of observed acoustic feature vectors assigned to state j , we could use the standard maximum likelihood estimates for the mean of the observation probability distribution:

$$\mu_j = \frac{\sum_{\mathbf{x} \in Z_j} \mathbf{x}}{|Z_j|}$$

Example



$$a_{11} = \frac{1}{2}$$
$$a_{12} = \frac{1}{2}$$

$$a_{22} = \frac{4}{5}$$
$$a_{23} = \frac{1}{5}$$

$$a_{33} = \frac{2}{3}$$
$$a_{3E} = \frac{1}{3}$$

- Viterbi training is an approximation—we would like to consider *all* possible paths

- Viterbi training is an approximation—we would like to consider *all* possible paths
- In this case rather than having a hard state-time alignment we estimate a probability

- Viterbi training is an approximation—we would like to consider *all* possible paths
- In this case rather than having a hard state-time alignment we estimate a probability
- *State occupation probability*: The probability $\gamma_j(t)$ of occupying state j at time t given the sequence of observations.

- Viterbi training is an approximation—we would like to consider *all* possible paths
- In this case rather than having a hard state-time alignment we estimate a probability
- *State occupation probability*: The probability $\gamma_j(t)$ of occupying state j at time t given the sequence of observations.
- We can use this for an iterative algorithm for HMM training: the EM algorithm

- Viterbi training is an approximation—we would like to consider *all* possible paths
- In this case rather than having a hard state-time alignment we estimate a probability
- *State occupation probability*: The probability $\gamma_j(t)$ of occupying state j at time t given the sequence of observations.
- We can use this for an iterative algorithm for HMM training: the EM algorithm
- Application of EM algorithm to HMMs is called '*Baum-Welch algorithm*'

If we have some initial parameters λ_0 and we want to find new parameters to maximise the likelihood $F_{\text{ML}}(\lambda)$, then we can instead maximise

$$\sum_{Q \in \mathcal{Q}} P(Q|X, \mathcal{M}, \lambda_0) \log P(X, Q|\mathcal{M}, \lambda)$$

- E-step** estimate the state occupation probabilities given the current parameters (Expectation)
- M-step** re-estimate the HMM parameters based on the estimated state occupation probabilities (Maximisation)

Backward probabilities

- To estimate the state occupation probabilities it is useful to define (recursively) another set of probabilities—the *Backward probabilities*

$$\beta_j(t) = p(x_{t+1}, \dots, x_T | q_t = j, \mathcal{M})$$

The probability of future observations given a the HMM is in state j at time t

Backward probabilities

- To estimate the state occupation probabilities it is useful to define (recursively) another set of probabilities—the *Backward probabilities*

$$\beta_j(t) = p(x_{t+1}, \dots, x_T | q_t = j, \mathcal{M})$$

The probability of future observations given a the HMM is in state j at time t

- These can be recursively computed (going backwards in time)

Backward probabilities

- To estimate the state occupation probabilities it is useful to define (recursively) another set of probabilities—the *Backward probabilities*

$$\beta_j(t) = p(x_{t+1}, \dots, x_T | q_t = j, \mathcal{M})$$

The probability of future observations given a the HMM is in state j at time t

- These can be recursively computed (going backwards in time)
 - Initialisation

$$\beta_i(T) = a_{iE}$$

Backward probabilities

- To estimate the state occupation probabilities it is useful to define (recursively) another set of probabilities—the *Backward probabilities*

$$\beta_j(t) = p(x_{t+1}, \dots, x_T | q_t = j, \mathcal{M})$$

The probability of future observations given a the HMM is in state j at time t

- These can be recursively computed (going backwards in time)
 - Initialisation

$$\beta_i(T) = a_{iE}$$

- Recursion

$$\beta_i(t) = \sum_{j=1}^J a_{ij} b_j(x_{t+1}) \beta_j(t+1) \quad \text{for } t = T-1, \dots, 1$$

Backward probabilities

- To estimate the state occupation probabilities it is useful to define (recursively) another set of probabilities—the *Backward probabilities*

$$\beta_j(t) = p(x_{t+1}, \dots, x_T | q_t = j, \mathcal{M})$$

The probability of future observations given a the HMM is in state j at time t

- These can be recursively computed (going backwards in time)
 - Initialisation

$$\beta_i(T) = a_{iE}$$

- Recursion

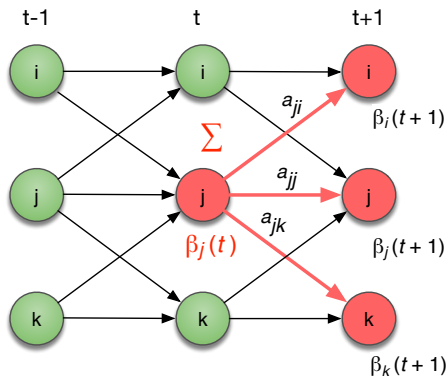
$$\beta_i(t) = \sum_{j=1}^J a_{ij} b_j(x_{t+1}) \beta_j(t+1) \quad \text{for } t = T-1, \dots, 1$$

- Termination

$$p(X | \mathcal{M}) = \beta_0(0) = \sum_{j=1}^J a_{0j} b_j(x_1) \beta_j(1) = \alpha_E$$

Backward Recursion

$$\beta_j(t) = p(x_{t+1}, \dots, x_T | q_t = j, \mathcal{M}) = \sum_{j=1}^J a_{ij} b_j(x_{t+1}) \beta_j(t+1)$$



State Occupation Probability

- The **state occupation probability** $\gamma_j(t)$ is the probability of occupying state j at time t given the sequence of observations
- Express in terms of the forward and backward probabilities:

$$\gamma_j(t) = P(q_t = j | X, \mathcal{M}) = \frac{1}{\alpha_E} \alpha_j(t) \beta_j(t)$$

recalling that $p(X | \mathcal{M}) = \alpha_E$

- Since

$$\begin{aligned} \alpha_j(t) \beta_j(t) &= p(x_1, \dots, x_t, q_t = j | \mathcal{M}) \\ &\quad p(x_{t+1}, \dots, x_T | q_t = j, \mathcal{M}) \\ &= p(x_1, \dots, x_t, x_{t+1}, \dots, x_T, q_t = j | \mathcal{M}) \\ &= p(X, q_t = j | \mathcal{M}) \end{aligned}$$

$$P(q_t = j | X, \mathcal{M}) = \frac{p(X, q_t = j | \mathcal{M})}{p(X | \mathcal{M})}$$

Re-estimation of transition probabilities

- Similarly to the state occupation probability, we can estimate $\xi_{i,j}(t)$, the probability of being in i at time t and j at $t + 1$, given the observations:

$$\begin{aligned}\xi_t(i, j) &= P(q_t = i, q_{t+1} = j | X, \mathcal{M}) \\ &= \frac{p(q_t = i, q_{t+1} = j, X | \mathcal{M})}{p(X | \mathcal{M})} \\ &= \frac{\alpha_i(t) a_{ij} b_j(x_{t+1}) \beta_j(t+1)}{\alpha_E}\end{aligned}$$

- We can use this to re-estimate the transition probabilities

$$\hat{a}_{ij} = \frac{\sum_{t=1}^T \xi_{i,j}(t)}{\sum_{k=1}^J \sum_{t=1}^T \xi_{i,k}(t)}$$

- See next lecture for re-estimation of observation probabilities $b_j(x)$

Pulling it all together

- Iterative estimation of HMM parameters using the EM algorithm. At each iteration
 - E step For all time-state pairs
 - 1 Recursively compute the forward probabilities $\alpha_j(t)$ and backward probabilities $\beta_j(t)$
 - 2 Compute the state occupation probabilities $\gamma_j(t)$ and $\xi_{i,j}(t)$
 - M step Based on the estimated state occupation probabilities re-estimate the HMM parameters: transition probabilities a_{ij} and parameters of the observation probabilities, $b_j(x)$
- The application of the EM algorithm to HMM training is sometimes called the Forward-Backward algorithm or Baum-Welch algorithm

Extension to a corpus of utterances

- We usually train from a large corpus of R utterances
- If x_t^r is the t th frame of the r th utterance X^r then we can compute the probabilities $\alpha_j^r(t)$, $\beta_j^r(t)$, $\gamma_j^r(t)$ and $\xi_{i,j}^r(t)$ as before
- The re-estimates are as before, except we must sum over the R utterances, ie:

$$\hat{a}_{ij} = \frac{\sum_{r=1}^R \sum_{t=1}^T \xi_{i,j}(t)}{\sum_{r=1}^R \sum_{k=1}^J \sum_{t=1}^T \xi_{i,k}(t)}$$

Extension to a corpus of utterances

- We usually train from a large corpus of R utterances
- If x_t^r is the t th frame of the r th utterance X^r then we can compute the probabilities $\alpha_j^r(t)$, $\beta_j^r(t)$, $\gamma_j^r(t)$ and $\xi_{i,j}^r(t)$ as before
- The re-estimates are as before, except we must sum over the R utterances, ie:

$$\hat{a}_{ij} = \frac{\sum_{r=1}^R \sum_{t=1}^T \xi_{i,j}(t)}{\sum_{r=1}^R \sum_{k=1}^J \sum_{t=1}^T \xi_{i,k}(t)}$$

- In addition, we usually employ “*embedded training*”, in which fine tuning of phone labelling with “*forced Viterbi alignment*” or forced alignment is involved. (For details see Section 9.7 in Jurafsky and Martin’s SLP)

Summary: HMMs

- HMMs provide a generative model for statistical speech recognition
- Three key problems
 - ① Computing the overall likelihood: the Forward algorithm
 - ② Decoding the most likely state sequence: the Viterbi algorithm
 - ③ Estimating the most likely parameters: the EM (Forward-Backward) algorithm
- Solutions to these problems are tractable due to the two key HMM assumptions
 - ① Conditional independence of observations given the current state
 - ② Markov assumption on the states

- Gales and Young (2007). “The Application of Hidden Markov Models in Speech Recognition”, *Foundations and Trends in Signal Processing*, **1** (3), 195–304: section 2.2.
- Jurafsky and Martin (2008). *Speech and Language Processing* (2nd ed.): sections 6.1–6.5; 9.2; 9.4. (Errata at <http://www.cs.colorado.edu/~martin/SLP/Errata/SLP2-PIEV-Errata.html>)
- Rabiner and Juang (1989). “An introduction to hidden Markov models”, *IEEE ASSP Magazine*, **3** (1), 4–16.
- Renals and Hain (2010). “Speech Recognition”, *Computational Linguistics and Natural Language Processing Handbook*, Clark, Fox and Lappin (eds.), Blackwells.