Hidden Markov Models

Steve Renals

Automatic Speech Recognition— ASR Lecture 5
2 February 2009

Overview

Fundamentals of HMMs

Today

- Statistical Speech Recognition
- HMM Acoustic Models
- Forward algorithm
- Viterbi algorithm

Overview

Fundamentals of HMMs

Today

- Statistical Speech Recognition
- HMM Acoustic Models
- Forward algorithm
- Viterbi algorithm

Thursday

- Forward-backward training
- Extension to mixture models

Variability in speech recognition

Several sources of variation

Size Number of word types in vocabulary, perplexity

Variability in speech recognition

Several sources of variation

Size Number of word types in vocabulary, perplexity

Style Continuously spoken or isolated? Planned monologue or spontaneous conversation?

Variability in speech recognition

Several sources of variation

- **Size** Number of word types in vocabulary, perplexity
- **Style** Continuously spoken or isolated? Planned monologue or spontaneous conversation?
- **Speaker** Tuned for a particular speaker, or speaker-independent? Adaptation to speaker characteristics and accent

Variability in speech recognition

Several sources of variation

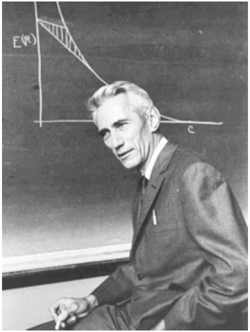
- **Size** Number of word types in vocabulary, perplexity
- **Style** Continuously spoken or isolated? Planned monologue or spontaneous conversation?
- **Speaker** Tuned for a particular speaker, or speaker-independent? Adaptation to speaker characteristics and accent
- **Acoustic environment** Noise, competing speakers, channel conditions (microphone, phone line, room acoustics)

Linguistic Knowledge or Machine Learning

- Intense effort needed to derive and encode linguistic rules that cover all the language
- Very difficult to take account of the variability of spoken language with such approaches
- Data-driven machine learning: Construct simple models of speech which can be learned from large amounts of data (thousands of hours of speech recordings)

Statistical Speech Recognition

А. А. Марков (1886).



Fundamental Equation of Statistical Speech Recognition

If $\mathbf{X}$ is the sequence of acoustic feature vectors (observations) and $\mathbf{W}$ denotes a word sequence, the most likely word sequence $\mathbf{W}^*$ is given by

$$\mathbf{W}^* = \arg\max_{\mathbf{W}} P(\mathbf{W} \mid \mathbf{X})$$

Fundamental Equation of Statistical Speech Recognition

If $\mathbf{X}$ is the sequence of acoustic feature vectors (observations) and $\mathbf{W}$ denotes a word sequence, the most likely word sequence $\mathbf{W}^*$ is given by

$$\mathbf{W}^* = \arg\max_{\mathbf{W}} P(\mathbf{W} \mid \mathbf{X})$$

Applying Bayes' Theorem:

$$\begin{aligned} P(\mathbf{W} \mid \mathbf{X}) &= \frac{p(\mathbf{X} \mid \mathbf{W}) P(\mathbf{W})}{p(\mathbf{X})} \\ &\propto p(\mathbf{X} \mid \mathbf{W}) P(\mathbf{W}) \\ \mathbf{W}^* &= \arg\max_{\mathbf{W}} \underbrace{p(\mathbf{X} \mid \mathbf{W})}_{\text{Acoustic model}} \quad \underbrace{P(\mathbf{W})}_{\text{Language model}} \end{aligned}$$

Statistical speech recognition

Statistical models offer a statistical "guarantee" — see the licence conditions of the best known automatic dictation system, for example:

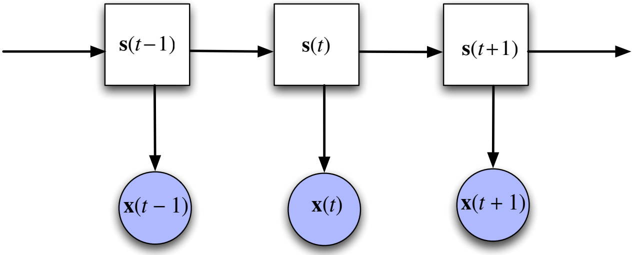
> *Licensee understands that* **speech recognition is a statistical process** *and that* **recognition errors are inherent in the process**. *licensee acknowledges that it is licensee s responsibility to* **correct recognition errors before using the results of the recognition**.

Hidden Markov Models





HMM Acoustic Model

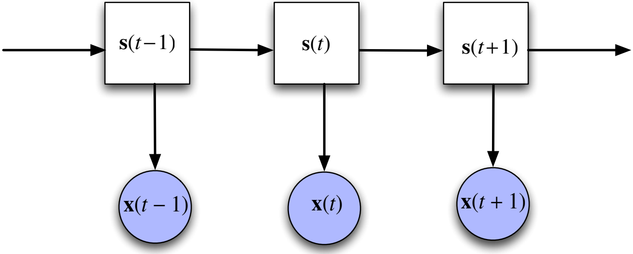


Hidden state **s** and observed acoustic features **x**

$$p(\mathbf{X} \mid \mathbf{W}) = \sum_{\mathbf{Q}} p(\mathbf{X} \mid \mathbf{Q}) P(\mathbf{Q} \mid \mathbf{W})$$

Q is a sequence of pronunciations

HMM Acoustic Model

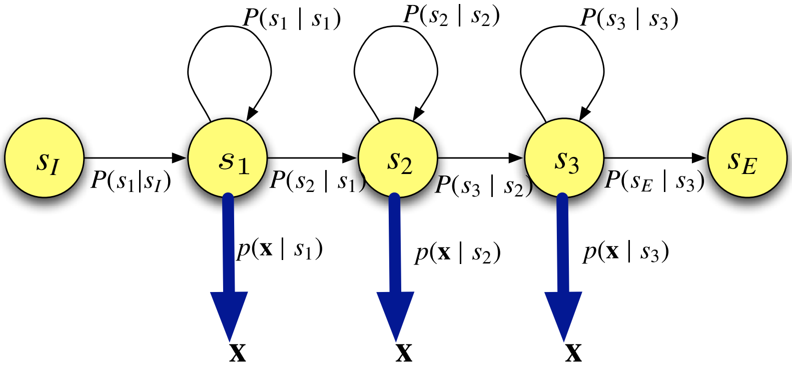


Hidden state **s** and observed acoustic features **x**

$$p(\mathbf{X} \mid \mathbf{W}) = \max_{\mathbf{Q}} p(\mathbf{X} \mid \mathbf{Q}) P(\mathbf{Q} \mid \mathbf{W})$$

Q is a sequence of pronunciations

Continuous Density HMM

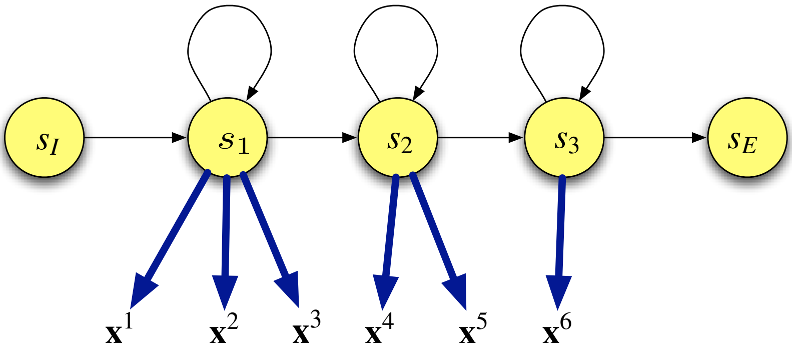


Probabilistic finite state automaton

Paramaters $\boldsymbol{\lambda}$:

- Transition probabilities: $a_{kj} = P(s_j \mid s_k)$
- Output probability density function: $b_j(\mathbf{x}) = p(\mathbf{x} \mid s_j)$

Continuous Density HMM

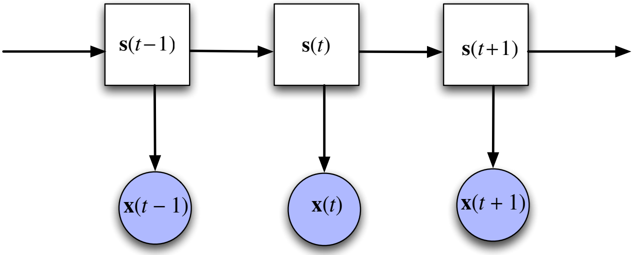


Probabilistic finite state automaton

Paramaters $\boldsymbol{\lambda}$:

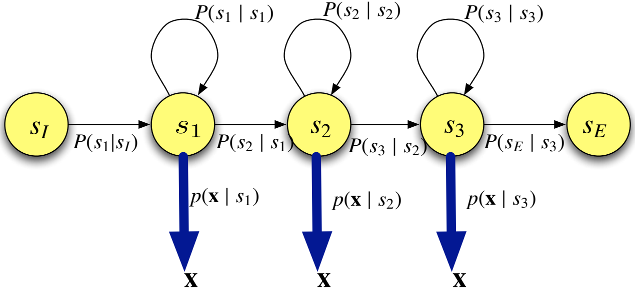
- Transition probabilities: $a_{kj} = P(s_j \mid s_k)$
- Output probability density function: $b_j(\mathbf{x}) = p(\mathbf{x} \mid s_j)$

HMM Assumptions

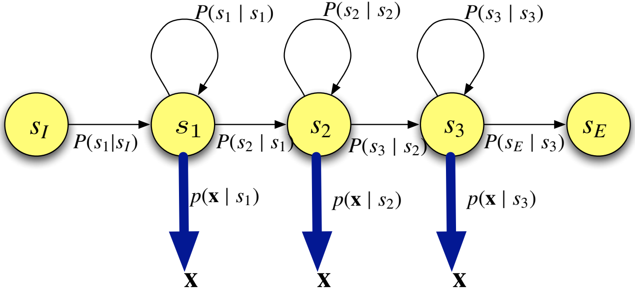


1. **Observation independence** An acoustic observation **x** is conditionally independent of all other observations given the state that generated it
2. **Markov process** A state is conditionally independent of all other states given the previous state

Output distribution



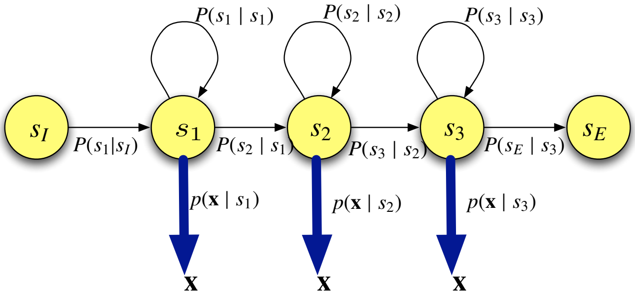
Output distribution



Single multivariate Gaussian with mean $\boldsymbol{\mu}^j$, covariance matrix $\boldsymbol{\Sigma}^j$:

$$b_j(\mathbf{x}) = p(\mathbf{x} \mid s_j) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}^j, \boldsymbol{\Sigma}^j)$$

Output distribution



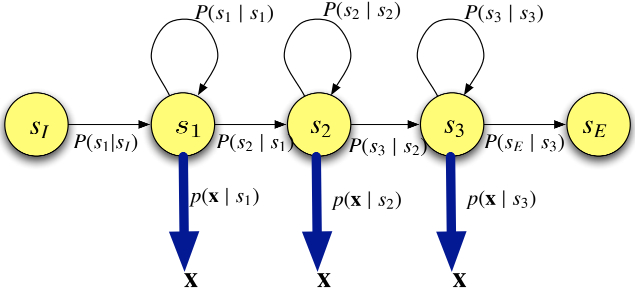
Single multivariate Gaussian with mean $\boldsymbol{\mu}^j$, covariance matrix $\boldsymbol{\Sigma}^j$:

$$b_j(\mathbf{x}) = p(\mathbf{x} \mid s_j) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}^j, \boldsymbol{\Sigma}^j)$$

M-component Gaussian mixture model:

$$b_j(\mathbf{x}) = p(\mathbf{x} \mid s_j) = \sum_{m=1}^{M} c_{jm}\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}^{jm}, \boldsymbol{\Sigma}^{jm})$$

Output distribution



Single multivariate Gaussian with mean $\boldsymbol{\mu}^j$, covariance matrix $\boldsymbol{\Sigma}^j$:

$$b_j(\mathbf{x}) = p(\mathbf{x} \mid s_j) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}^j, \boldsymbol{\Sigma}^j)$$

M-component Gaussian mixture model:

$$b_j(\mathbf{x}) = p(\mathbf{x} \mid s_j) = \sum_{m=1}^{M} c_{jm} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}^{jm}, \boldsymbol{\Sigma}^{jm})$$

The three problems of HMMs

Working with HMMs requires the solution of three problems:

1. **Likelihood** Determine the overall likelihood of an observation sequence $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$ being generated by an HMM

The three problems of HMMs

Working with HMMs requires the solution of three problems:

1. **Likelihood** Determine the overall likelihood of an observation sequence $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$ being generated by an HMM
2. **Decoding** Given an observation sequence and an HMM, determine the most probable hidden state sequence

The three problems of HMMs

Working with HMMs requires the solution of three problems:

1. **Likelihood** Determine the overall likelihood of an observation sequence $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$ being generated by an HMM
2. **Decoding** Given an observation sequence and an HMM, determine the most probable hidden state sequence
3. **Training** Given an observation sequence and an HMM, learn the best HMM parameters $\boldsymbol{\lambda} = \{\{a_{jk}\}, \{b_j()\}\}$

1. Likelihood: The Forward algorithm

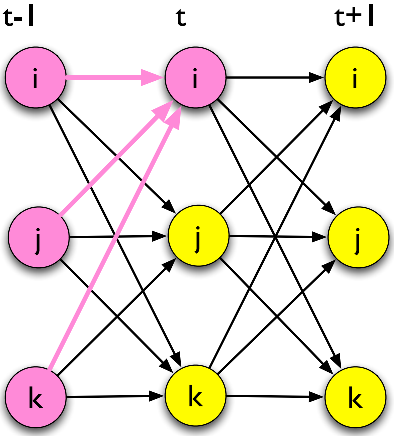
- Goal: determine $p(\mathbf{X} \mid \boldsymbol{\lambda})$
- Sum over all possible state sequences $s_1 s_2 \ldots s_T$ that could result in the observation sequence $\mathbf{X}$

1. Likelihood: The Forward algorithm

- Goal: determine $p(\mathbf{X} \mid \boldsymbol{\lambda})$
- Sum over all possible state sequences $s_1 s_2 \ldots s_T$ that could result in the observation sequence $\mathbf{X}$
- Rather than enumerating each sequence, compute the probabilities recursively (exploiting the Markov assumption)

Recursive algorithms on HMMs

Visualize the problem as a *state-time trellis*



1. Likelihood: The Forward algorithm

- Goal: determine $p(\mathbf{X} \mid \boldsymbol{\lambda})$
- Sum over all possible state sequences $s_1 s_2 \ldots s_T$ that could result in the observation sequence $\mathbf{X}$
- Rather than enumerating each sequence, compute the probabilities recursively (exploiting the Markov assumption)

1. Likelihood: The Forward algorithm

- Goal: determine $p(\mathbf{X} \mid \boldsymbol{\lambda})$
- Sum over all possible state sequences $s_1 s_2 \ldots s_T$ that could result in the observation sequence $\mathbf{X}$
- Rather than enumerating each sequence, compute the probabilities recursively (exploiting the Markov assumption)
- *Forward probability*, $\alpha_t(s_j)$: the probability of observing the observation sequence $\mathbf{x}_1 \ldots \mathbf{x}_t$ and being in state s_j at time t:

$$\alpha_t(s_j) = p(\mathbf{x}_1, \ldots, \mathbf{x}_t, S(t) = s_j \mid \boldsymbol{\lambda})$$

1. Likelihood: The Forward recursion

- Initialization

$$\alpha_0(s_I) = 1$$
$$\alpha_0(s_j) = 0 \qquad \text{if } s_j \neq s_I$$

1. Likelihood: The Forward recursion

- Initialization

$$\alpha_0(s_I) = 1$$
$$\alpha_0(s_j) = 0 \qquad \text{if } s_j \neq s_I$$

- Recursion

$$\alpha_t(s_j) = \sum_{i=1}^{N} \alpha_{t-1}(s_i) a_{ij} b_j(\mathbf{x}_t)$$

1. Likelihood: The Forward recursion

- Initialization

$$\alpha_0(s_I) = 1$$
$$\alpha_0(s_j) = 0 \qquad \text{if } s_j \neq s_I$$

- Recursion

$$\alpha_t(s_j) = \sum_{i=1}^{N} \alpha_{t-1}(s_i) a_{ij} b_j(\mathbf{x}_t)$$

- Termination

$$p(\mathbf{X} \mid \boldsymbol{\lambda}) = \alpha_T(s_E) = \sum_{i=1}^{N} \alpha_T(s_i) a_{iE}$$

Interim Summary

- Framework for statistical speech recognition
- HMM acoustic models
- HMM likelihood computation: the Forward algorithm
- Reading
  - Jurafsky and Martin (2008). *Speech and Language Processing*(2nd ed.): sections 6.1–6.5; 9.2; 9.4.
  - 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.
  - Rabiner and Juang (1989). "An introduction to hidden Markov models", *IEEE ASSP Magazine*, **3** (1), 4–16.

Hidden Markov Models (part 2)

Steve Renals

Automatic Speech Recognition— ASR Lecture 6
5 February 2009

Overview

Fundamentals of HMMs

Previously

- Statistical Speech Recognition
- HMM Acoustic Models
- Forward algorithm

Today

- Viterbi algorithm
- Forward-backward training
- Extension to mixture models

Overview

Fundamentals of HMMs

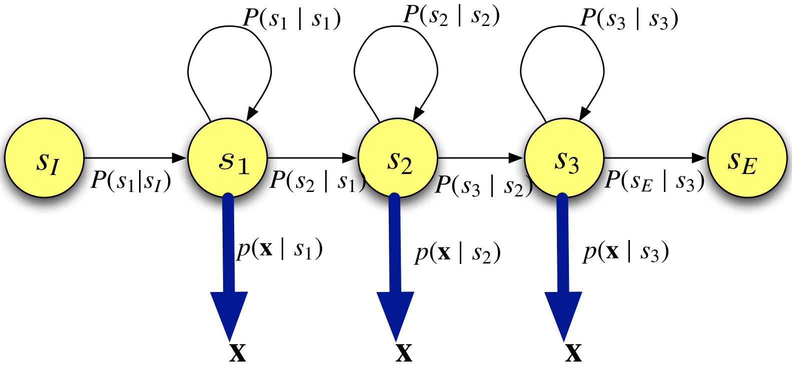
Previously

- Statistical Speech Recognition
- HMM Acoustic Models
- Forward algorithm

Today

- Viterbi algorithm
- Forward-backward training
- Extension to mixture models

Continuous Density HMM

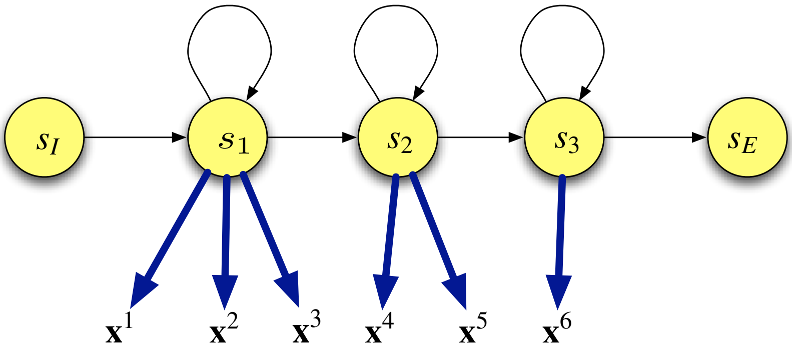


Probabilistic finite state automaton

Paramaters $\boldsymbol{\lambda}$:

- Transition probabilities: $a_{kj} = P(s_j \mid s_k)$
- Output probability density function: $b_j(\mathbf{x}) = p(\mathbf{x} \mid s_j)$

Continuous Density HMM



Probabilistic finite state automaton

Paramaters $\boldsymbol{\lambda}$:

- Transition probabilities: $a_{kj} = P(s_j \mid s_k)$
- Output probability density function: $b_j(\mathbf{x}) = p(\mathbf{x} \mid s_j)$

The three problems of HMMs

Working with HMMs requires the solution of three problems:

1. **Likelihood** Determine the overall likelihood of an observation sequence $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$ being generated by an HMM

The three problems of HMMs

Working with HMMs requires the solution of three problems:

1. **Likelihood** Determine the overall likelihood of an observation sequence $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$ being generated by an HMM
2. **Decoding** Given an observation sequence and an HMM, determine the most probable hidden state sequence

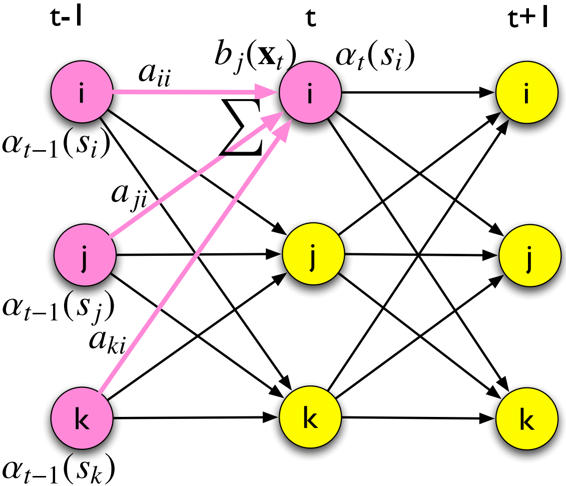
The three problems of HMMs

Working with HMMs requires the solution of three problems:

1. **Likelihood** Determine the overall likelihood of an observation sequence $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_t, \ldots, \mathbf{x}_T)$ being generated by an HMM
2. **Decoding** Given an observation sequence and an HMM, determine the most probable hidden state sequence
3. **Training** Given an observation sequence and an HMM, learn the best HMM parameters $\boldsymbol{\lambda} = \{\{a_{jk}\}, \{b_j()\}\}$

1. Likelihood: Forward Recursion

$$\alpha_t(s_j) = p(\mathbf{x}_1, \ldots, \mathbf{x}_t, S(t) = s_j \mid \boldsymbol{\lambda})$$



Viterbi approximation

- Instead of summing over all possible state sequences, just consider the most likely

Viterbi approximation

- Instead of summing over all possible state sequences, just consider the most likely
- Achieve this by changing the summation to a maximisation in the recursion:

$$V_t(s_j) = \max_i V_{t-1}(s_i) a_{ij} b_j(\mathbf{x}_t)$$

Viterbi approximation

- Instead of summing over all possible state sequences, just consider the most likely
- Achieve this by changing the summation to a maximisation in the recursion:

$$V_t(s_j) = \max_i V_{t-1}(s_i) a_{ij} b_j(\mathbf{x}_t)$$

- Changing the recursion in this way gives the likelihood of the most probable path

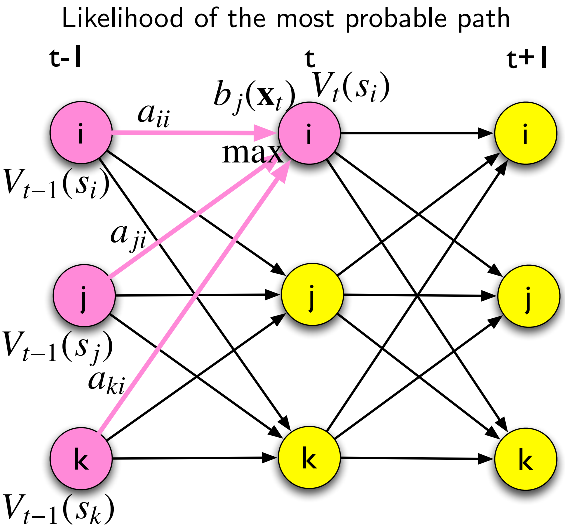
Viterbi approximation

- Instead of summing over all possible state sequences, just consider the most likely
- Achieve this by changing the summation to a maximisation in the recursion:

$$V_t(s_j) = \max_i V_{t-1}(s_i) a_{ij} b_j(\mathbf{x}_t)$$

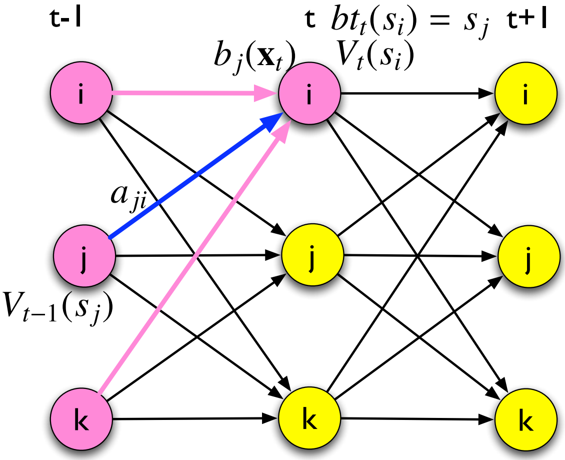
- Changing the recursion in this way gives the likelihood of the most probable path
- We need to keep track of the states that make up this path by keeping a sequence of *backpointers* to enable a Viterbi *backtrace*: the backpointer for each state at each time indicates the previous state on the most probable path

Viterbi Recursion



Viterbi Recursion

Backpointers to the previous state on the most probable path



2. Decoding: The Viterbi algorithm

- Initialization

$$
\begin{aligned}
V_0(s_I) &= 1 \\
V_0(s_j) &= 0 \qquad \text{if } s_j \neq s_I \\
bt_0(s_j) &= 0
\end{aligned}
$$

2. Decoding: The Viterbi algorithm

- Initialization

$$V_0(s_I) = 1$$
$$V_0(s_j) = 0 \qquad \text{if } s_j \neq s_I$$
$$bt_0(s_j) = 0$$

- Recursion

$$V_t(s_j) = \max_{i=1}^{N} V_{t-1}(s_i) a_{ij} b_j(\mathbf{x}_t)$$
$$bt_t(s_j) = \arg\max_{i=1}^{N} V_{t-1}(s_i) a_{ij} b_j(\mathbf{x}_t)$$

2. Decoding: The Viterbi algorithm

- Initialization

$$V_0(s_I) = 1$$
$$V_0(s_j) = 0 \qquad \text{if } s_j \neq s_I$$
$$bt_0(s_j) = 0$$

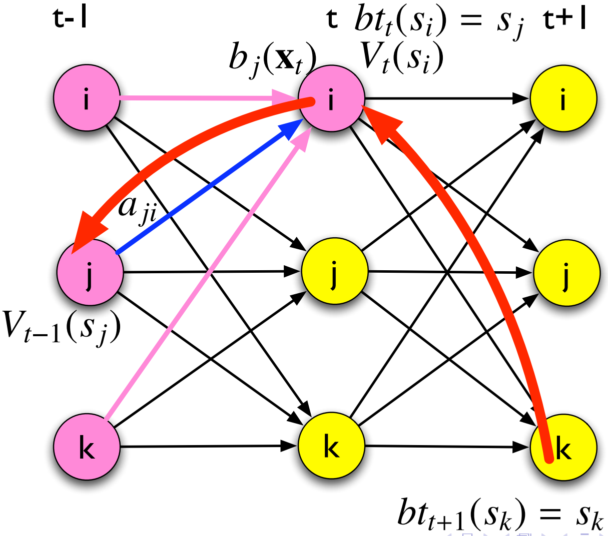
- Recursion

$$V_t(s_j) = \max_{i=1}^{N} V_{t-1}(s_i) a_{ij} b_j(\mathbf{x}_t)$$
$$bt_t(s_j) = \arg\max_{i=1}^{N} V_{t-1}(s_i) a_{ij} b_j(\mathbf{x}_t)$$

- Termination

$$P^* = V_T(s_E) = \max_{i=1}^{N} V_T(s_i) a_{iE}$$
$$s_T^* = bt_T(q_E) = \arg\max_{i=1}^{N} V_T(s_i) a_{iE}$$

Backtrace to find the state sequence of the most probable path



3. Training: Forward-Backward algorithm

- Goal: Efficiently estimate the parameters of an HMM $\boldsymbol{\lambda}$ from an observation sequence

3. Training: Forward-Backward algorithm

- Goal: Efficiently estimate the parameters of an HMM $\boldsymbol{\lambda}$ from an observation sequence
- Assume single Gaussian output probability distribution

$$b_j(\mathbf{x}) = p(\mathbf{x} \mid s_j) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}^j, \boldsymbol{\Sigma}^j)$$

3. Training: Forward-Backward algorithm

- Goal: Efficiently estimate the parameters of an HMM $\boldsymbol{\lambda}$ from an observation sequence
- Assume single Gaussian output probability distribution

$$b_j(\mathbf{x}) = p(\mathbf{x} \mid s_j) = \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}^j, \boldsymbol{\Sigma}^j)$$

- Parameters $\boldsymbol{\lambda}$:
  - Transition probabilities a_{ij}:

$$\sum_i a_{ij} = 1$$

  - Gaussian parameters for state s_j:
    mean vector $\boldsymbol{\mu}^{\mathbf{j}}$; covariance matrix $\boldsymbol{\Sigma}^{\mathbf{j}}$

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 $C(s_i \rightarrow s_j)$ is the count of transitions from s_i to s_j

$$\hat{a}_{ij} = \frac{C(s_i \rightarrow s_j)}{\sum_k C(s_k \rightarrow s_j)}$$

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 $C(s_i \rightarrow s_j)$ is the count of transitions from s_i to s_j

$$\hat{a}_{ij} = \frac{C(s_i \rightarrow s_j)}{\sum_k C(s_k \rightarrow s_j)}$$

- Likewise if Z_j is the set of observed acoustic feature vectors assigned to state j, we can use the standard maximum likelihood estimates for the mean and the covariance:

$$\hat{\boldsymbol{\mu}}^j = \frac{\sum_{\mathbf{x} \in Z_j} \mathbf{x}}{|Z_j|}$$

$$\hat{\boldsymbol{\Sigma}}^j = \frac{\sum_{\mathbf{x} \in Z_j} (\mathbf{x} - \hat{\boldsymbol{\mu}}^j)(\mathbf{x} - \hat{\boldsymbol{\mu}}^j)^T}{|Z_j|}$$

EM Algorithm

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

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

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 $\gamma_t(s_j)$ of occupying state s_j at time t given the sequence of observations

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 $\gamma_t(s_j)$ of occupying state s_j at time t given the sequence of observations
- We can use this for an iterative algorithm for HMM training: the EM algorithm

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 $\gamma_t(s_j)$ of occupying state s_j at time t given the sequence of observations
- We can use this for an iterative algorithm for HMM training: the EM algorithm
- Each iteration has two steps:

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 $\gamma_t(s_j)$ of occupying state s_j at time t given the sequence of observations
- We can use this for an iterative algorithm for HMM training: the EM algorithm
- Each iteration has two steps:

  E-step estimate the state occupation probabilities (Expectation)

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 $\gamma_t(s_j)$ of occupying state s_j at time t given the sequence of observations
- We can use this for an iterative algorithm for HMM training: the EM algorithm
- Each iteration has two steps:

  E-step estimate the state occupation probabilities (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

$$\beta_t(s_j) = p(\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \mathbf{x}_T \mid S(t) = s_j, \boldsymbol{\lambda})$$

The probability of future observations given a the HMM is in state s_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

  $$\beta_t(s_j) = p(\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \mathbf{x}_T \mid S(t) = s_j, \boldsymbol{\lambda})$$

  The probability of future observations given a the HMM is in state s_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

  $$\beta_t(s_j) = p(\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \mathbf{x}_T \mid S(t) = s_j, \boldsymbol{\lambda})$$

  The probability of future observations given a the HMM is in state s_j at time t
- These can be recursively computed (going backwards in time)
  - Initialisation

    $$\beta_T(s_i) = a_{iE}$$

Backward probabilities

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

$$\beta_t(s_j) = p(\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \mathbf{x}_T \mid S(t) = s_j, \boldsymbol{\lambda})$$

  The probability of future observations given a the HMM is in state s_j at time t
- These can be recursively computed (going backwards in time)
  - Initialisation

$$\beta_T(s_i) = a_{iE}$$

  - Recursion

$$\beta_t(s_i) = \sum_{j=1}^{N} a_{ij} b_j(\mathbf{x}_{t+1}) \beta_{t+1}(s_j)$$

Backward probabilities

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

  $$\beta_t(s_j) = p(\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \mathbf{x}_T \mid S(t) = s_j, \boldsymbol{\lambda})$$

  The probability of future observations given a the HMM is in state s_j at time t
- These can be recursively computed (going backwards in time)
  - Initialisation

    $$\beta_T(s_i) = a_{iE}$$

  - Recursion

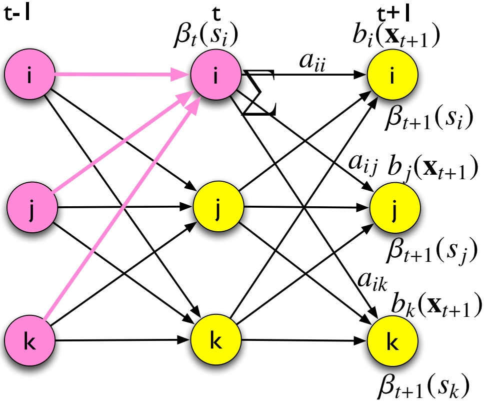
    $$\beta_t(s_i) = \sum_{j=1}^{N} a_{ij} b_j(\mathbf{x}_{t+1}) \beta_{t+1}(s_j)$$

  - Termination

    $$p(\mathbf{X} \mid \boldsymbol{\lambda}) = \beta_0(s_I) = \sum_{j=1}^{N} a_{Ij} b_j(\mathbf{x}_1) \beta_1(s_j) = \alpha_T(s_E)$$

Backward Recursion

$$\beta_t(s_j) = p(\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \mathbf{x}_T \mid S(t) = s_j, \boldsymbol{\lambda})$$



State Occupation Probability

- The **state occupation probability** $\gamma_t(s_j)$ is the probability of occupying state s_j at time t given the sequence of observations

State Occupation Probability

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

$$\gamma_t(s_j) = P(S(t) = s_j \mid \mathbf{X}, \boldsymbol{\lambda}) = \frac{1}{\alpha_T(s_E)} \alpha_t(j) \beta_t(j)$$

recalling that $p(\mathbf{X}|\boldsymbol{\lambda}) = \alpha_T(s_E)$

State Occupation Probability

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

$$\gamma_t(s_j) = P(S(t) = s_j \mid \mathbf{X}, \boldsymbol{\lambda}) = \frac{1}{\alpha_T(s_E)} \alpha_t(j)\beta_t(j)$$

recalling that $p(\mathbf{X}|\boldsymbol{\lambda}) = \alpha_T(s_E)$

- Since

$$\begin{aligned}
\alpha_t(s_j)\beta_t(s_j) &= p(\mathbf{x}_1, \ldots, \mathbf{x}_t, S(t) = s_j \mid \boldsymbol{\lambda}) \\
&\qquad p(\mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \mathbf{x}_T \mid S(t) = s_j, \boldsymbol{\lambda}) \\
&= p(\mathbf{x}_1, \ldots, \mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \ldots, \mathbf{x}_T, S(t) = s_j \mid \boldsymbol{\lambda}) \\
&= p(\mathbf{X}, S(t) = s_j \mid \boldsymbol{\lambda})
\end{aligned}$$

$$P(S(t) = s_j \mid \mathbf{X}, \boldsymbol{\lambda}) = \frac{p(\mathbf{X}, S(t) = s_j \mid \boldsymbol{\lambda})}{p(\mathbf{X}|\boldsymbol{\lambda})}$$

Re-estimation of Gaussian parameters

- The sum of state occupation probabilities through time for a state, may be regarded as a "soft" count

Re-estimation of Gaussian parameters

- The sum of state occupation probabilities through time for a state, may be regarded as a "soft" count
- We can use this "soft" alignment to re-estimate the HMM parameters:

$$\hat{\boldsymbol{\mu}}^j = \frac{\sum_{t=1}^{T} \gamma_t(s_j)\mathbf{x}_t}{\sum_{t=1}^{T} \gamma_t(s_j)}$$

$$\hat{\boldsymbol{\Sigma}}^j = \frac{\sum_{t=1}^{T} \gamma_t(s_j)(\mathbf{x}_t - \hat{\boldsymbol{\mu}}^j)(\mathbf{x} - \hat{\boldsymbol{\mu}}^j)^T}{\sum_{t=1}^{T} \gamma_t(s_j)}$$

Re-estimation of transition probabilities

- Similarly to the state occupation probability, we can estimate $\xi_t(s_i, s_j)$, the probability of being in s_i at time t and s_j at $t+1$, given the observations:

$$\begin{aligned}\xi_t(s_i, s_j) &= P(S(t) = s_i, S(t+1) = s_j \mid \mathbf{X}, \boldsymbol{\lambda}) \\ &= \frac{P(S(t) = s_i, S(t+1) = s_j, \mathbf{X} \mid \boldsymbol{\lambda})}{p(\mathbf{X}|\Lambda)} \\ &= \frac{\alpha_t(s_i) a_{ij} b_j(\mathbf{x}_{t+1}) \beta_{t+1}(s_j)}{\alpha_T(s_E)}\end{aligned}$$

Re-estimation of transition probabilities

- Similarly to the state occupation probability, we can estimate $\xi_t(s_i, s_j)$, the probability of being in s_i at time t and s_j at $t+1$, given the observations:

$$\begin{aligned}\xi_t(s_i, s_j) &= P(S(t) = s_i, S(t+1) = s_j \mid \mathbf{X}, \boldsymbol{\lambda}) \\ &= \frac{P(S(t) = s_i, S(t+1) = s_j, \mathbf{X} \mid \boldsymbol{\lambda})}{p(\mathbf{X}|\Lambda)} \\ &= \frac{\alpha_t(s_i) a_{ij} b_j(\mathbf{x}_{t+1}) \beta_{t+1}(s_j)}{\alpha_T(s_E)}\end{aligned}$$

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

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \xi_t(s_i, s_j)}{\sum_{k=1}^{N} \sum_{t=1}^{T} \xi_t(s_i, s_k)}$$

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 $\alpha_t(s_j)$ and backward probabilities $\beta_t(j)$
  2. Compute the state occupation probabilities $\gamma_t(s_j)$ and $\xi_t(s_i, s_j)$

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 $\alpha_t(s_j)$ and backward probabilities $\beta_t(j)$
  2. Compute the state occupation probabilities $\gamma_t(s_j)$ and $\xi_t(s_i, s_j)$

  M step Based on the estimated state occupation probabilities re-estimate the HMM parameters: mean vectors $\boldsymbol{\mu}^j$, covariance matrices $\boldsymbol{\Sigma}^j$ and transition probabilities a_{ij}

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 $\alpha_t(s_j)$ and backward probabilities $\beta_t(j)$
  2. Compute the state occupation probabilities $\gamma_t(s_j)$ and $\xi_t(s_i, s_j)$

  M step Based on the estimated state occupation probabilities re-estimate the HMM parameters: mean vectors $\boldsymbol{\mu}^j$, covariance matrices $\boldsymbol{\Sigma}^j$ and transition probabilities a_{ij}

- The application of the EM algorithm to HMM training is sometimes called the Forward-Backward algorithm

Extension to a corpus of utterances

- We usually train from a large corpus of R utterances
- If $\mathbf{x}_t^r$ is the tth frame of the rth utterance $\mathbf{X}^r$ then we can compute the probabilities $\alpha_t^r(j)$, $\beta_t^r(j)$, $\gamma_t^r(s_j)$ and $\xi_t^r(s_i, s_j)$ as before
- The re-estimates are as before, except we must sum over the R utterances, eg:

$$\hat{\boldsymbol{\mu}}^j = \frac{\sum_{r=1}^{R} \sum_{t=1}^{T} \gamma_t^r(s_j)\mathbf{x}_t^r}{\sum_{r=1}^{R} \sum_{t=1}^{T} \gamma_t^r(s_j)}$$

Extension to Gaussian mixture model (GMM)

- The assumption of a Gaussian distribution at each state is very strong; in practice the acoustic feature vectors associated with a state may be strongly non-Gaussian

Extension to Gaussian mixture model (GMM)

- The assumption of a Gaussian distribution at each state is very strong; in practice the acoustic feature vectors associated with a state may be strongly non-Gaussian
- In this case an M-component Gaussian mixture model is an appropriate density function:

$$b_j(\mathbf{x}) = p(\mathbf{x} \mid s_j) = \sum_{m=1}^{M} c_{jm} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}^{jm}, \boldsymbol{\Sigma}^{jm})$$

Given enough components, this family of functions can model any distribution.

Extension to Gaussian mixture model (GMM)

- The assumption of a Gaussian distribution at each state is very strong; in practice the acoustic feature vectors associated with a state may be strongly non-Gaussian
- In this case an M-component Gaussian mixture model is an appropriate density function:

$$b_j(\mathbf{x}) = p(\mathbf{x} \mid s_j) = \sum_{m=1}^{M} c_{jm} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}^{jm}, \boldsymbol{\Sigma}^{jm})$$

  Given enough components, this family of functions can model any distribution.
- Train using the EM algorithm, in which the component estimation probabilities are estimated in the E-step

EM training of HMM/GMM

- Rather than estimating the state-time alignment, we estimate the component/state-time alignment, and component-state occupation probabilities $\gamma_t(s_j, m)$: the probability of occupying mixture component m of state s_j at time t

EM training of HMM/GMM

- Rather than estimating the state-time alignment, we estimate the component/state-time alignment, and component-state occupation probabilities $\gamma_t(s_j, m)$: the probability of occupying mixture component m of state s_j at time t
- We can thus re-estimate the mean of mixture component m of state s_j as follows

$$\hat{\boldsymbol{\mu}}^{jm} = \frac{\sum_{t=1}^{T} \gamma_t(s_j, m)\mathbf{x}_t}{\sum_{t=1}^{T} \gamma_t(s_j, m)}$$

And likewise for the covariance matrices (mixture models often use diagonal covariance matrices)

EM training of HMM/GMM

- Rather than estimating the state-time alignment, we estimate the component/state-time alignment, and component-state occupation probabilities $\gamma_t(s_j, m)$: the probability of occupying mixture component m of state s_j at time t
- We can thus re-estimate the mean of mixture component m of state s_j as follows

$$\hat{\boldsymbol{\mu}}^{jm} = \frac{\sum_{t=1}^{T} \gamma_t(s_j, m)\mathbf{x}_t}{\sum_{t=1}^{T} \gamma_t(s_j, m)}$$

  And likewise for the covariance matrices (mixture models often use diagonal covariance matrices)
- The mixture coefficients are re-estimated in a similar way to transition probabilities:

$$\hat{c}_{jm} = \frac{\sum_{t=1}^{T} \gamma_t(s_j, m)}{\sum_{\ell=1}^{M} \sum_{t=1}^{T} \gamma_t(s_j, \ell)}$$

Doing the computation

- The forward, backward and Viterbi recursions result in a long sequence of probabilities being multiplied
- This can cause floating point *underflow* problems
- In practice computations are performed in the log domain (in which multiplies become adds)
- Working in the log domain also avoids needing to perform the exponentiation when computing Gaussians

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

References: HMMs

- 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)
- 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.
- Rabiner and Juang (1989). "An introduction to hidden Markov models", *IEEE ASSP Magazine*, **3** (1), 4–16.