Doctoral Course in Speech Recognition

Friday March 30

Mats Blomberg

March-June 2007

General course info

- Home page
  - http://www.speech.kth.se/~matsb/speech_speaker_rec_course_2007/Course_PM.html
- Exercises
  - VQ, CART, HMM decoding, training
  - Return solutions by May 7
- Term paper
  - 4 - 6 pages, max 10
  - Send to reviewers (2 course participants) by May 16
  - Reviewer return comments by May 25
  - Final paper to teacher and the reviewers by June 1
- Closing seminar
  - Presentation of own paper
  - Active discussions

Course overview

- Day #1
  - Probability, Statistics and Information Theory (pp 73-131: 59 pages)
  - Pattern Recognition (pp 133-197: 65 pages)
  - Speech Signal Representations (pp 275-336 62 pages)
  - Hidden Markov Models (pp 377-413: 37 pages)
- Day #2
  - Hidden Markov Models (cont.)
  - Acoustic Modeling (pp 415-475: 61 pages)
  - Environmental Robustness (pp 477-544: 68 pages)
  - Computational exercise
- Day #3
  - Language Modeling (pp 545-590: 46 pages)
  - Basic Search Algorithms (pp 591-643: 53 pages)
  - Large-Vocabulary Search Algorithms (pp 645-685: 41 pages)
  - Applications and User Interfaces (pp 919-956: 38 pages)
  - Other topics
- Day #4 Closing seminar
  - Presentations of term papers

8.2.4 How to Estimate HMM Parameters - Baum-Welch Algorithm

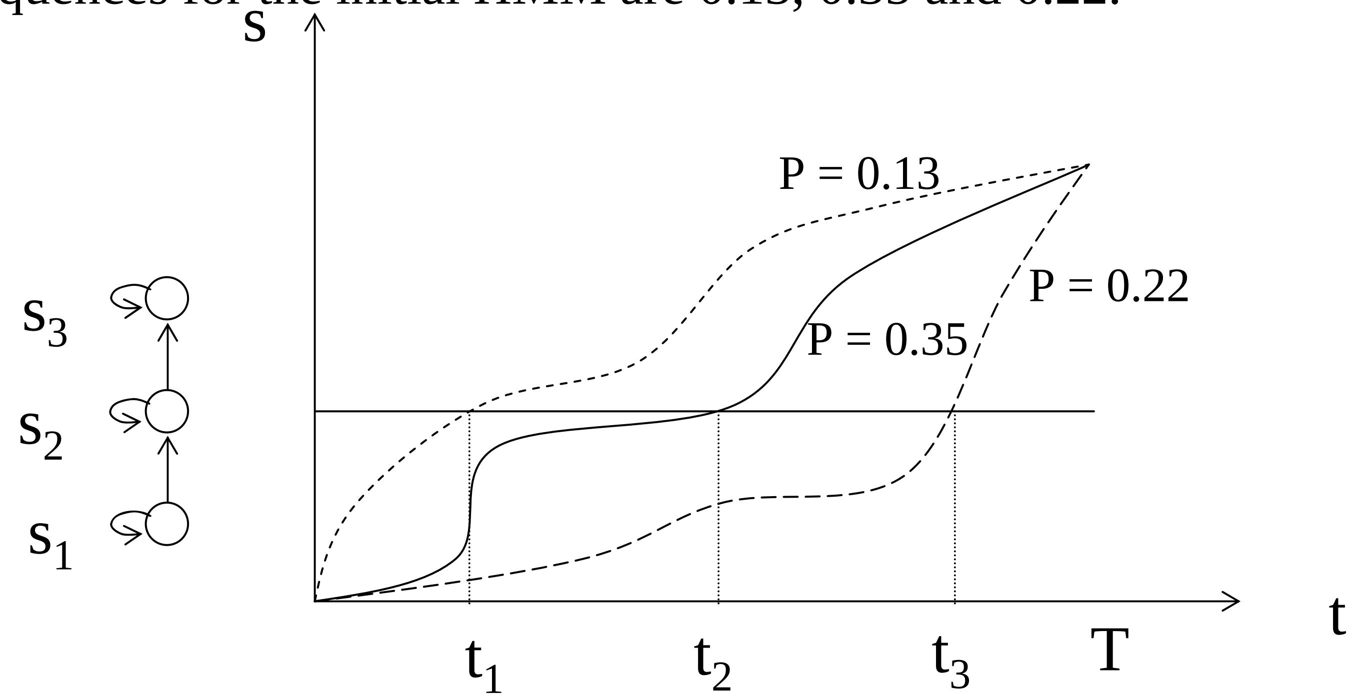
- The most difficult of the three HMM problems
- Unsupervised learning. Incomplete data. State sequence unknown.
  - Use the EM algorithm
- Implemented by the iterative Baum-Welch (Forward-Backward) algorithm

The EM Algorithm for HMM training

- Problem approached
  - Estimate model parameters that maximize the probability that the model have generated the data.
    - The maximum is achieved when the model distribution is equal to that of the the training data
  - Estimate distributions (ML) of several classes (model states) when the training data (time frames) are not classified
  - Is it possible to train the classes anyway? (Yes - *local* maximum only)
- Iterative procedure, simplified description
  - 1. Initialise class distributions
  - 2. Using current parameters, compute the class (state) probabilities for each training sample (time frame)
  - 3. Every class (state) distribution is re-computed as a probability weighted contribution of each sample (time frame)
    - A moving target; the new distribution will affect the class probabilities, which will, in turn, result in new parameters, therefore:
  - 4. Repeat 2+3 until convergence (Will converge)
- Similar principle for observation and transition probabilities

Simplified illustration of EM estimation for HMM training

Say, three paths have been found. The probabilities of the state sequences for the initial HMM are 0.13, 0.35 and 0.22.



New $E(s_2) = (0.13\ X(t_1) + 0.35\ X\ (t_2) + 0.22\ X(t_3))\ /\ 0.70$

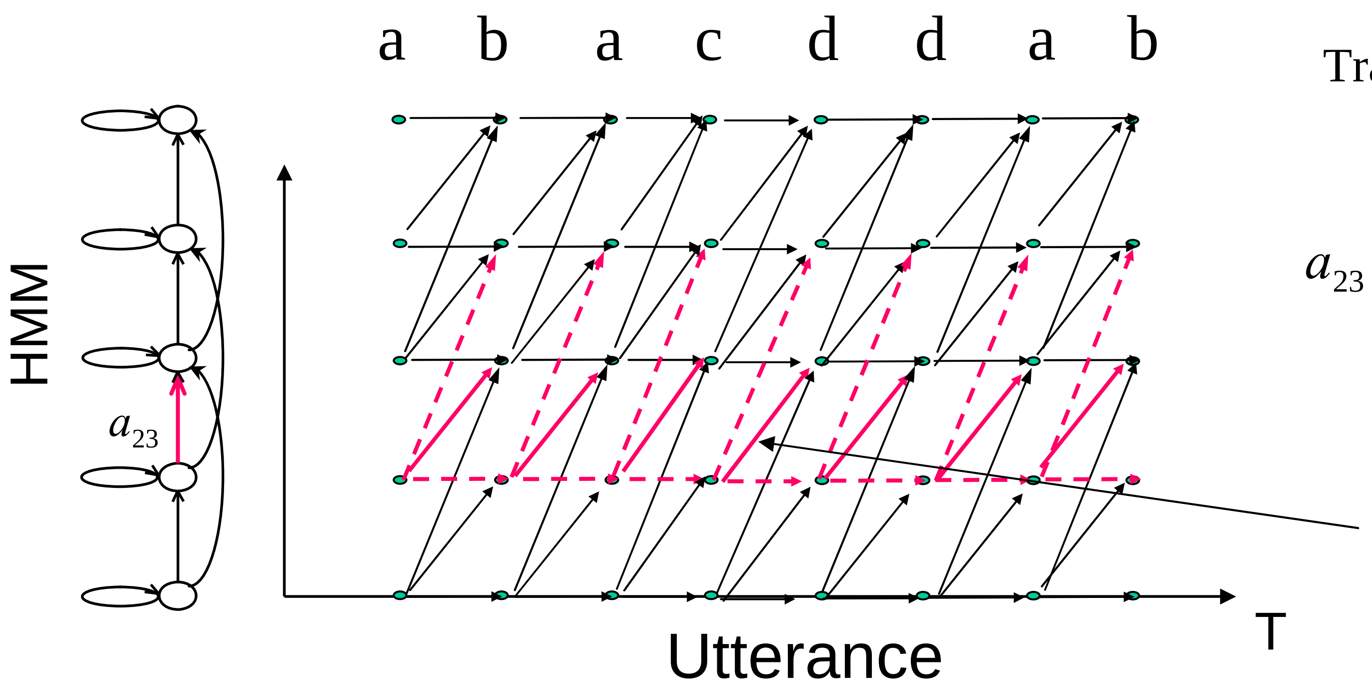
Not as simple as it may look. Many paths, partly shared

Towards Baum-Welch algorithm

- Not feasible to compute over all individual possible state sequences
- And not necessary
  - The probability that the model has taken a certain transition at a certain time is independent of the history and the future (Markov assumption)
- We only need to know their summed effect to the probability for every individual transition in the trellis diagram (time-state)
  - P(The model switches between states i and j at time t)

The Baum-Welch algorithm

P(The model switched from state *i* to *j* at time *t*) = $\gamma_t(i,j)$



Transition probability:

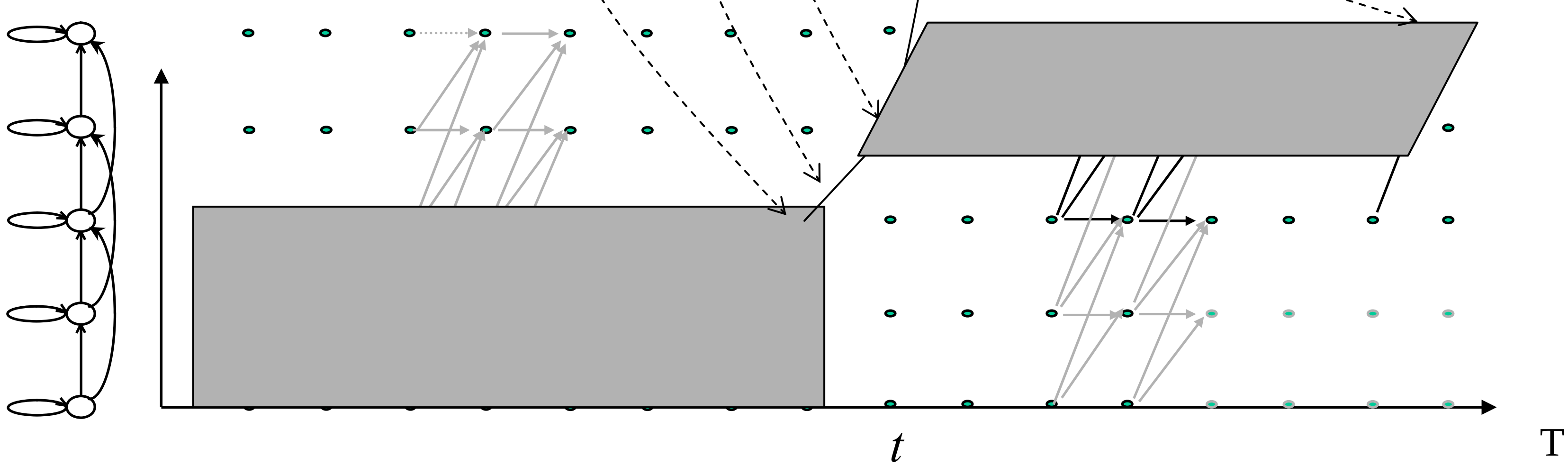
$$a_{23} = \frac{\sum_{t=1}^{8} \gamma_t(2,3)}{\sum_{t=1}^{8} \sum_{k=2}^{4} \gamma_t(2,k)}$$

$\gamma_4(2,3)$

The Baum-Welch Algorithm - 2

$\gamma_t(i,j)$: The probability of the model having taken the transition from state i to state j at time t and produced the observations
= The sum of the probabilities for all paths passing through (t-1,i)and (t,j) divided by the sum of the probabilities for all paths

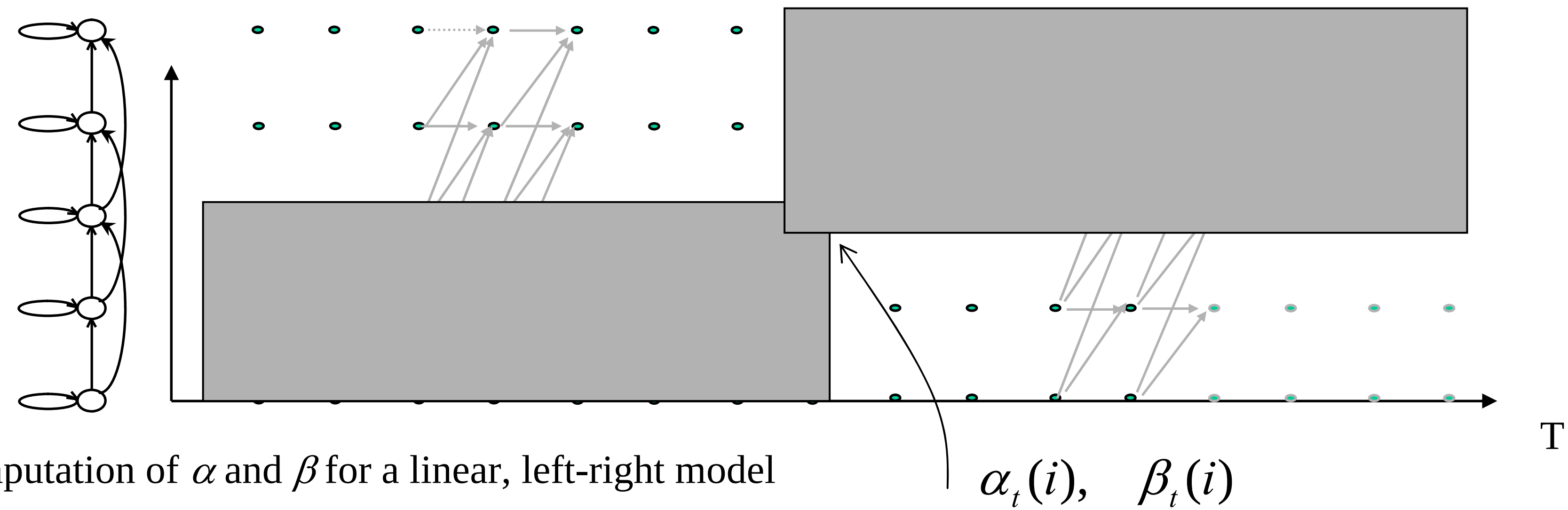
$$= \frac{\alpha_{t-1}(i) a_{ij} b_j(X_t) \beta_t(j)}{\sum_{k=1}^{N} \alpha_T(k)}$$



Forward and Backward probabilities

- Forward probability $\alpha_t(i)$ $\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}\right] b_j(X_t)$
  - The probability of generating a partial observation $X_1 \ldots X_t$ ending at time *t* and state *i*
- Backward probability $\beta_t(i)$
  - The probability of generating a partial observation $X_{t+1} \ldots X_T$ starting from time *t* and state *i*.

$$\beta_t(i) = \left[\sum_{j=1}^{N} a_{ij} b_j(X_{t+1}) \beta_{t+1}(j)\right] \quad t = T-1, \ldots, 1 \quad 1 \le i \le N \quad \beta_T(i) = 1/N$$



Computation of α and β for a linear, left-right model

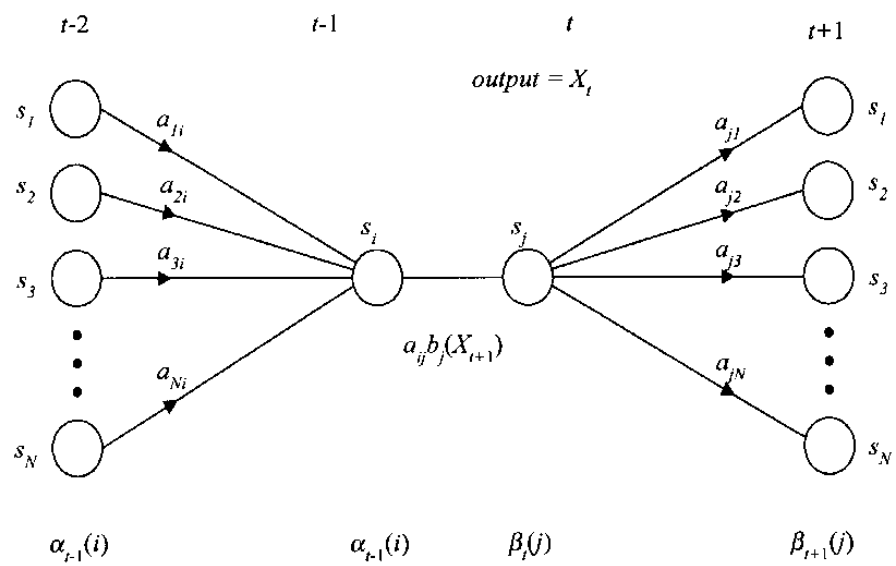
The Baum-Welch Algorithm (cont.)

Def $\gamma_t(i,j)$: The probability of the model having taken the transition from state i to state j at time t

$$\gamma_t(i,j) = P(\text{The model has switched from state } i \text{ to } j \text{ at time } t)$$

$$= \frac{\text{P(The model generates the observed sequence and switches from state i to j at time t)}}{\text{P(The model generates the observed sequence)}}$$

$$= \frac{\alpha_{t-1}(i) a_{ij} b_j(X_t) \beta_t(j)}{\sum_{k=1}^{N} \alpha_T(k)}$$



The Baum-Welch Algorithm (cont.)

New model estimates:

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \gamma_t(i,j)}{\sum_{t=1}^{T}\sum_{k=1}^{N} \gamma_t(i,k)} \qquad (8.40)$$

The ratio between the expected number of transitions from state *i* to *j* and the expected number of all transitions from state *i*

$$\hat{b}_j(k) = \frac{\sum_{t \in X_t = o_k} \sum_{i} \gamma_t(i,j)}{\sum_{t=1}^{T} \sum_{i} \gamma_t(i,j)} \qquad (8.41)$$

The ratio between the expected number of times the observation data emitted from state j is o_k and the expected number of times any observation data is emitted from state j

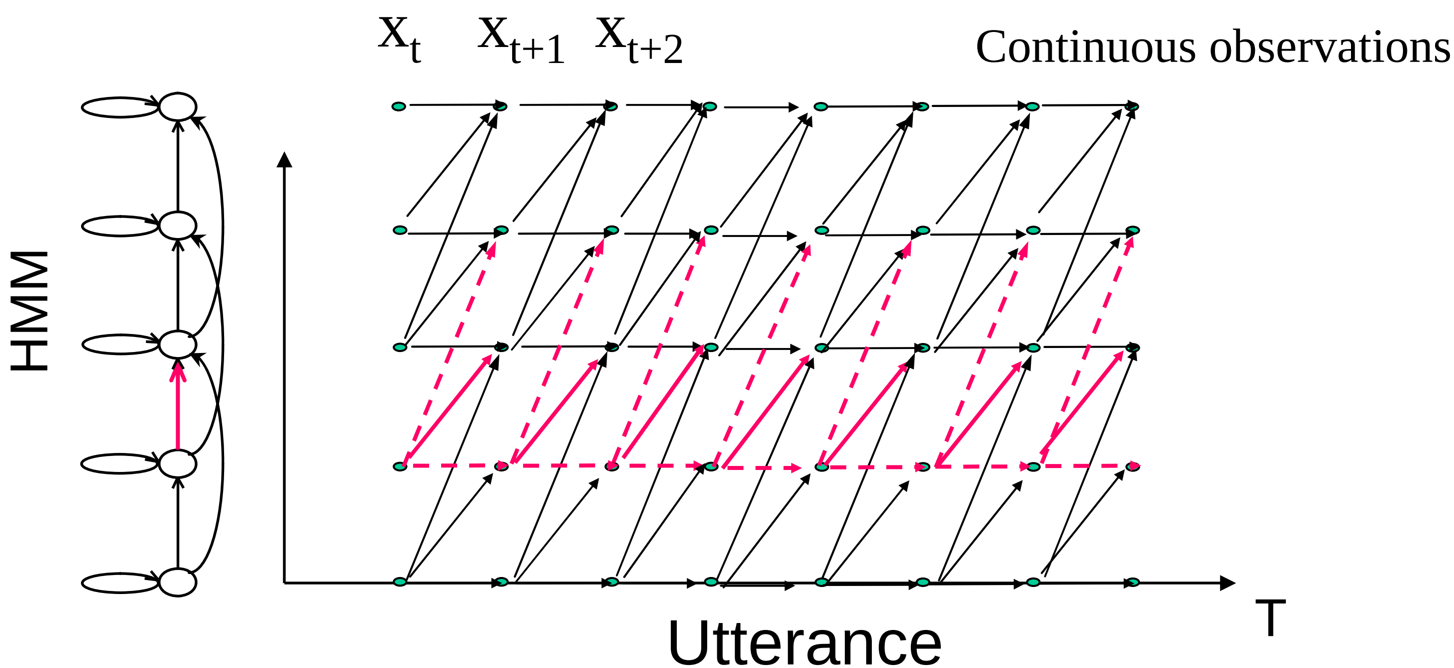
Quite intuitive equations!

8.3 Continuous and Semi-Continuous HMMs

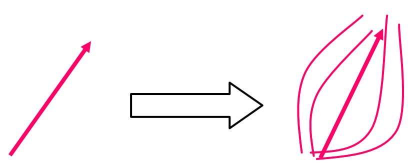
- The observation does not come from a finite set, but from a continuous space
  - No quantization error

The Baum-Welch algorithm
Continuous Mixture Density HMMs

P(The model switched to state *j* at time *t* using mixture component *k*) = $\zeta_t(j,k)$



General idea:



One arrow for each mixture component + P(symbol) => $N(\mu,\sigma)$

8.3.1 Continuous Mixture Density HMMs

- A weighted sum of multivariate Gaussians

$$b_j(\mathbf{x}) = \sum_{k=1}^{M} c_{jk} N(x, \mu_{jk}, \Sigma_{jk})$$

  - M: number of mixture-components
  - c_{jk} : the weight for the k^th component in state j
  - Similar re-estimation equations as the discrete case but an extra dimension is the probability of each mixture component having produced the observation

$$\sum_{k=1}^{M} c_{jk} = 1$$

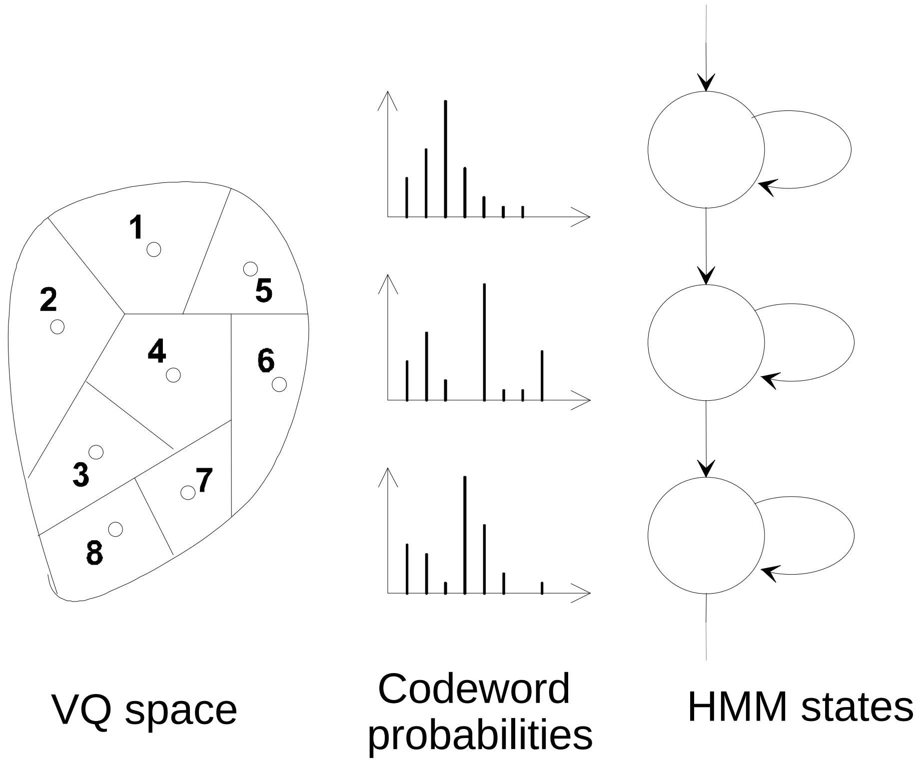
$$\hat{\boldsymbol{\mu}}_{jk} = \frac{\sum_{t=1}^{T} \zeta_t(j,k)\mathbf{x}_t}{\sum_{t=1}^{T} \zeta_t(j,k)} \qquad \hat{\Sigma}_{jk} = \frac{\sum_{t=1}^{T} \zeta_t(j,k)(x-\hat{\boldsymbol{\mu}}_{jk})(x-\hat{\boldsymbol{\mu}}_{jk})'}{\sum_{t=1}^{T} \zeta_t(j,k)} \qquad \hat{c}_{jk} = \frac{\sum_{t=1}^{T} \zeta_t(j,k)}{\sum_{t=1}^{T}\sum_{k=1}^{M} \zeta_t(j,k)}$$

$$\zeta_t(j,k) = \frac{\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} c_{jk} b_{jk}(x_t) \beta_t(j)}{\sum_{i=1}^{N} \alpha_T(i)}$$

Zeta: The probability that there is a transition to state j and mixture comp k, at time t given that the model generates the observed sequence

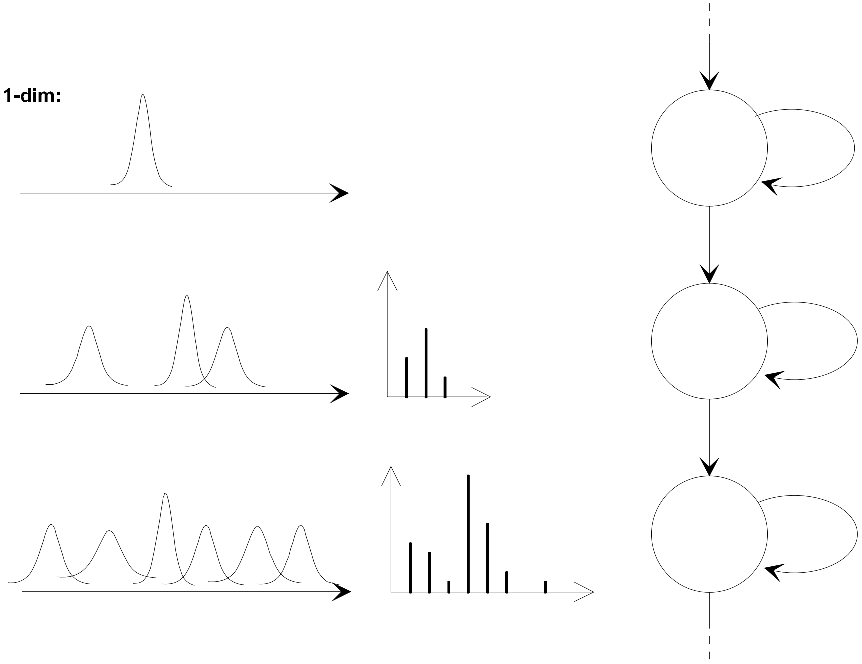
Discrete HMM

- Observations are discrete symbols, e.g. VQ codeword indeces



Continuous HMM

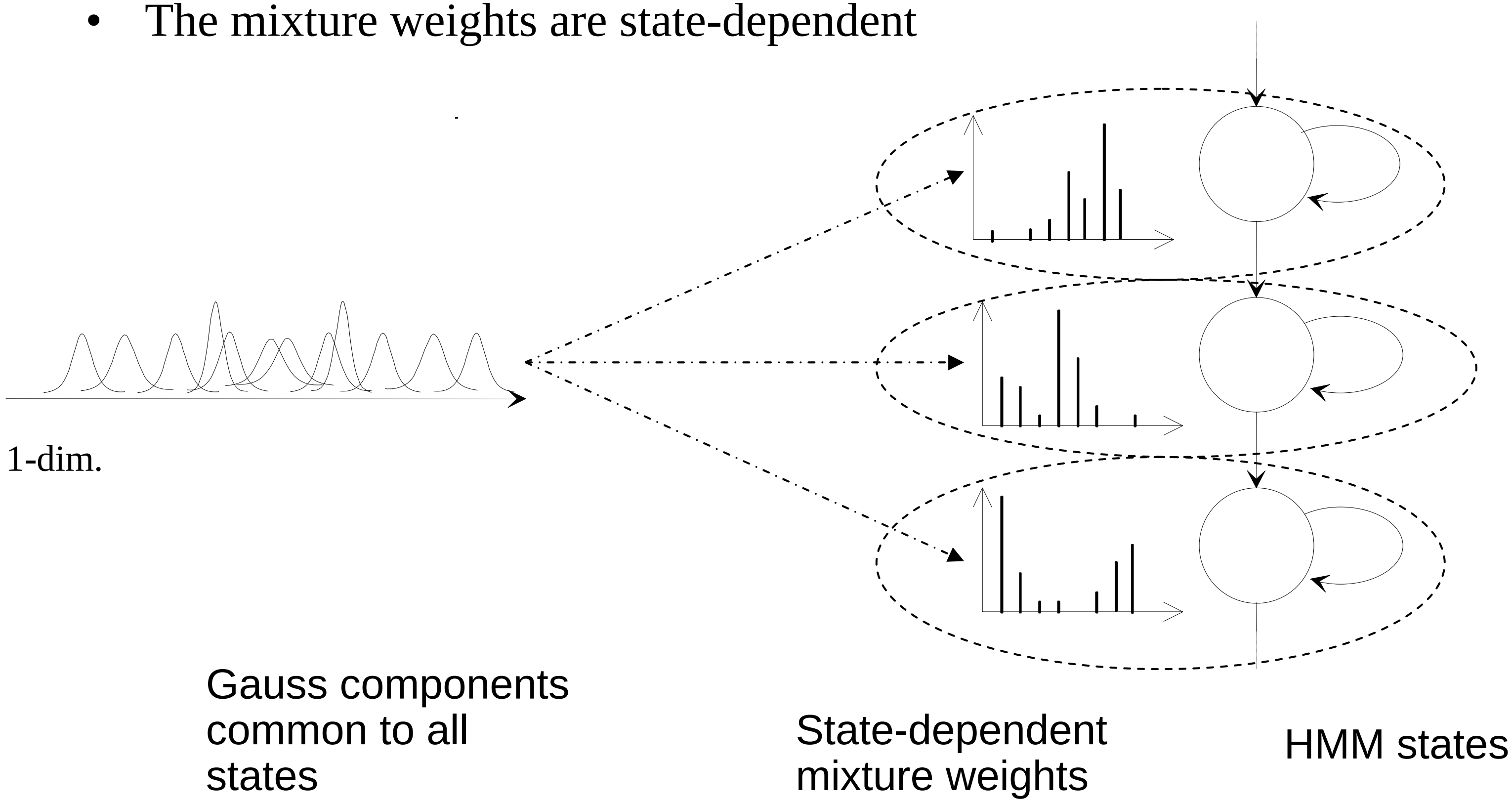
- Continuous observations
- The mixture components and the mixture weights are state-dependent



Gauss components   Mixture weights   HMM states

Semicontinuous (Tied-Mixture) HMM

- Continuous observations
- The mixture component distributions are state-independent
- The mixture weights are state-dependent



8.3.2 Semicontinuous HMMs (SCHMM)
(Tied-mixture HMM)

- Bridging the gap between discrete and continuous mixture density HMMs.
- As in discrete HMM, a common codebook for all states
  - But continuous pdfs of Gaussians - not discrete symbols
  - State-dependent mixture weights corresponds to discrete output probabilities b_j
- The observation probability is a sum of the individual probabilities for all mixture components.

$$b_j(\mathbf{x}) = \sum_{k=1}^{M} b_j(k) N(\mathbf{x}, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

- Reduced number of parameters (vs continuous) but still allowing detailed acoustic modeling (vs discrete)
- Similar re-estimation as for continuous HMM, except state-independent mixtures

8.4 Practical Issues in Using HMMs

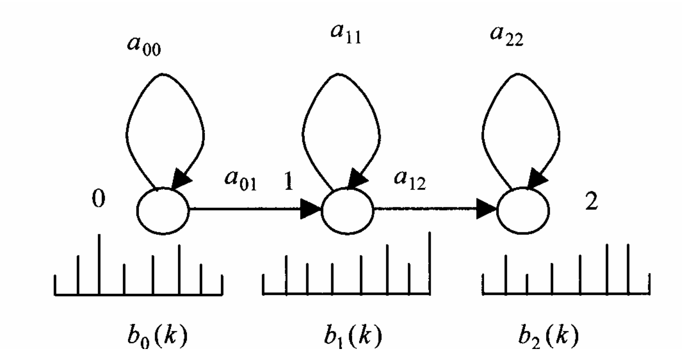
- Initial Estimates
- Model Topology
- Training Criteria
- Deleted Interpolation
- Parameter Smoothing
- Probability Representations

8.4.1 Initial Estimates

- Important in order to reach a high local maximum
- Discrete HMM
  - Initial zero probability remains zero
  - Uniform distribution works reasonably well
- Continuous HMM methods
  - *k*-means clustering
  - Proceed from discrete HMM to semi-continuous to continuous
  - Start training single mixture models.
- Use previously segmented data or “flat start” (equal model parameters of all states in the training data)

8.4.2 Model Topology

- Left-to-right - the most popular topology
  - Number of states per model
    - Phone: Three to five states
    - Short words: 2-3 states per phoneme
    - Long words: 1-2 states per phoneme



- Null transitions
  - Change state without using observations (e.g. initial and final states)

8.4.3 Training Criteria

- Maximum Likelihood Estimation (MLE)
  - Sensitive to inaccurate Markov assumptions
- MCE and MMIE might work better
- MAP suitable for adaptation and small training data
- MLE the most used
  - simplicity
  - superior performance
- MCE and MMIE for small and medium vocabularies
- Combinations possible

8.4.4 Deleted Interpolation

- Combine well-trained general models with less well-trained detailed models
  - The combined probability is a weighted average (interpolation) of the probabilities of the separate models
    - speaker-dependent and -independent models, context-free and context-dependent phone models, unigrams, bigrams and trigrams

$$P_{DI}(\mathbf{x}) = \lambda P_A(\mathbf{x}) + (1-\lambda) P_B(\mathbf{x})$$

  - V-fold cross-validation to estimate λ
    - Train on (v-1) parts, estimate λ on the remaining part, circulate
    - EM algorithm: new λ = λ P(model A) / (P(interpolated model DI)

Example: Combine trigrams, bigrams and unigrams

$$P(w_k \mid w_{k-2}, w_{k-1}) = \lambda_3 \frac{C(w_{k-2}, w_{k-1}, w_k)}{C(w_{k-2}, w_{k-1})} + \lambda_2 \frac{C(w_{k-1}, w_k)}{C(w_{k-1})} + \lambda_1 \frac{C(w_k)}{C(K)}$$

Alg. 8.5 Deleted Interpolation Procedure

- Step 1: Initialize λ with a guessed estimate
- Step 2: Update λ by the formula:

$$\hat{\lambda} = \frac{1}{M} \sum_{j=1}^{M} \sum_{t=1}^{n_j} \frac{\lambda P_{A-j}(\mathbf{x}_t^j)}{\lambda P_{A-j}(\mathbf{x}_t^j) + (1-\lambda) P_{B-j}(\mathbf{x}_t^j)}$$

  - M: number of training data divisions
  - P_{A-j} and P_{B-j} are estimated on the all training data except part j
  - n_j the number of data points in part j aligned to the model
  - $\mathbf{x}_t^j$: the t-th data point in the j-th set
- Step 3: Repeat step 2 until convergence

8.4.5 Parameter Smoothing
Compensate for insufficient training data

- Increase the data *(There is no data like more data)*
- Reduce the number of free parameters
- Deleted interpolation
- Parameter flooring to avoid small probability values
- Tying parameters (SCHMM)
- Covariance matrix
  – interpolate via MAP
  – Tie matrices
  – Use diagonal covariance matrices

8.4.6 Probability Representations

- The probabilities become very small
  - underflow problem
- Viterbi decoding (only multiplication): use logarithm
- Forward-backward (multiplication and addition): difficult
  - Solution 1
    - Scaling to make $\sum_i S_t \alpha_t(i) = 1$
  - Solution 2
    - Logarithmic probabilities
    - Use look-up table to speed up $\log(P_1+P_2)$

8.5 HMM Limitations

- Duration modeling
- First Order Assumption
- Conditional Independence Assumption

8.5.1 Duration Modeling

- HMM duration distribution: exponential decrease
  - The probability of state duration *t* is the probability of taking the self-loop t times multiplied by the probability of leaving the state

$$d_i(t) = a_{ii}^t (1 - a_{ii})$$

  - Different to real distribution (House & Crystal, 1986)
  - Maximum Likelihood search can model non-exponential distributions by sequence of states
    - Standard Viterbi cannot but can be modified to at the expense of large increase in computation

Modified Viterbi for state duration modelling

- Algorithm

$$V_t(j) = \underset{i}{Max}\,\underset{\tau}{Max}\left[V_{t-\tau}(i) a_{ij} d_j(\tau) \prod_{l=1}^{\tau} b_j(X_{t-\tau+l}) \right]$$

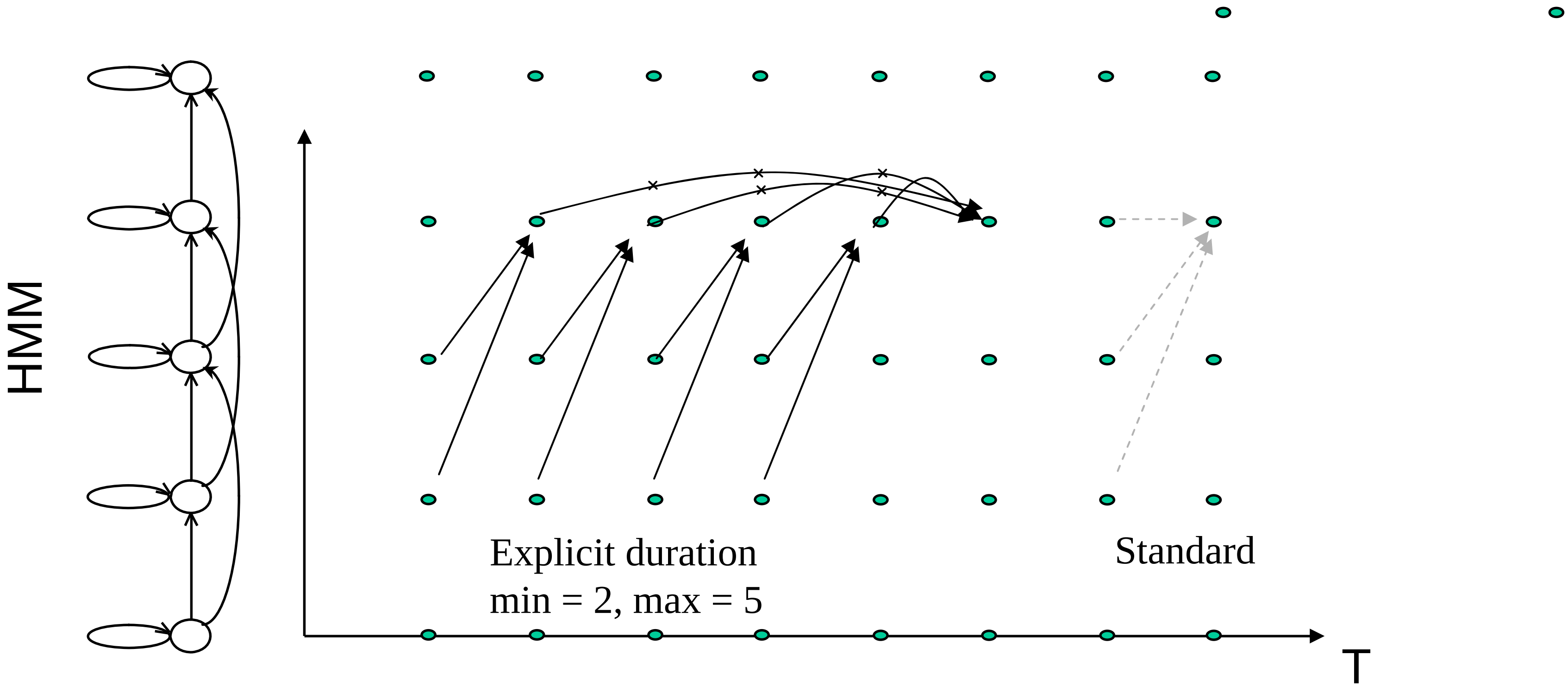
- Conventional Viterbi algorithm

$$V_t(j) = \underset{i}{Max}\left\lfloor V_{t-1}(i) a_{ij} b_j(X_t) \right\rfloor$$

- Maximize over previous state *i* and starting time t-τ for end time *t*
- Large increase in complexity $O(D^2)$, (D = Max duration)
  - Can be reduced (Seward, 2003)
- Modest accuracy improvement

Recursion pattern for explicit duration Viterbi decoding

$$V_t(j) = \underset{i}{Max}\,\underset{\tau}{Max}\left[V_{t-\tau}(i) a_{ij} d_j(\tau) \prod_{l=1}^{\tau} b_j(X_{t-\tau+l}) \right]$$



8.5.2 First Order Assumption

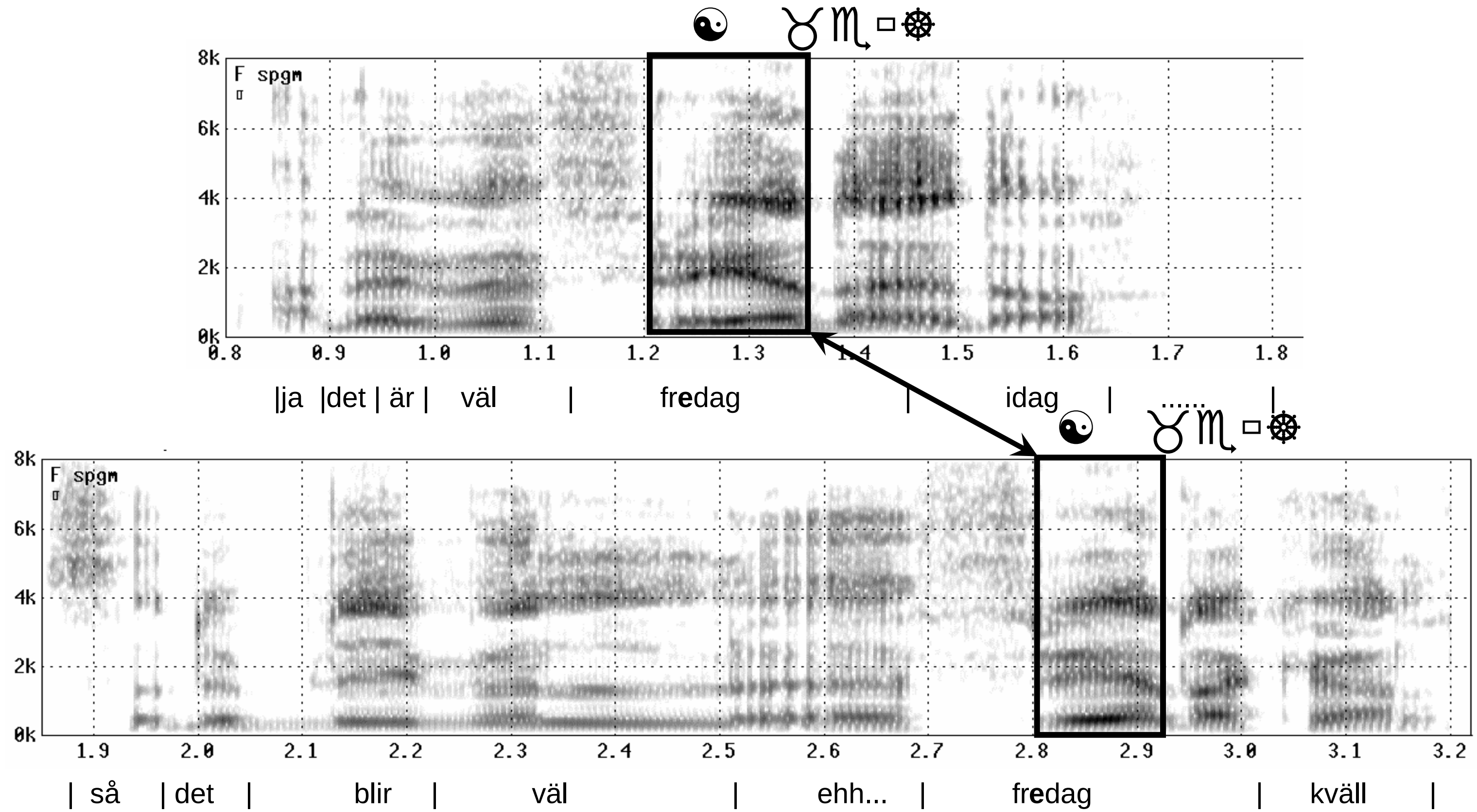
- In a first order Markov chain, the transition from a state depends only on the current state
- A second order Markov chain models the transition probability between two states as dependent on the current and the previous state
- Transition probability: $a_{s_{t-2}s_{t-1}s_t} = P(s_t | s_{t-2}s_{t-1})$

- Has not offered sufficient accuracy improvement to justify the increase in computational complexity

8.5.3 Conditional Independence Assumption

- It is assumed that all observation frames are dependent only on the state that generated them, not on any other frames in the observation sequence (neighboring or distant)
- Difficult to handle non-stationary frames with strong correlation
- To include the dependence on the previous observation frame:
- $$P(\mathbf{X}|\mathbf{S},\mathbf{\Phi}) = \prod_{t=1}^{T} P(X_t|X_{t-1}, s_t, \mathbf{\Phi})$$ or $$P(\mathbf{X}|\mathbf{S},\mathbf{\Phi}) = \prod_{t=1}^{T} P(X_t|\Re(X_{t-1}), s_t, \mathbf{\Phi})$$

Distant repetitions of identical phonemes in an utterance are acoustically similar



If the first [e:] has certain characteristics (due to accent, age, gender, etc.) then it is likely that the second one has it as well. Dependence! Problem in speaker-independent ASR

9.6 Adaptive Techniques – Minimizing Mismatches

- There is always mismatch between training and recognition conditions
- Adaptation
  - Minimize the mismatch dynamically with little calibration data
  - Supervised
    - knowledge of the correct identity
  - Unsupervised
    - the recognition result is assumed to be correct

9.6.1 Maximum a Posteriori (MAP)

- A new model is estimated using the training data interpolated with old information about the model

$$\hat{\mu}_{ik} = \frac{\tau_{ik}\mu_{nw_{ik}} + \sum_{t=1}^{T}\zeta_t(i,k)\mathbf{x}_t}{\tau_{ik} + \sum_{t=1}^{T}\zeta_t(i,k)}$$

- τ_{ik} is a balancing factor between the prior mean and the ML estimate. Can be a constant for all Gaussian components
- Similar for the covariance estimation
- Limitations
  - The prior model needs to be accurate
  - Needs observations for all models

9.6.2 Maximum Likelihood Linear Regression (MLLR)

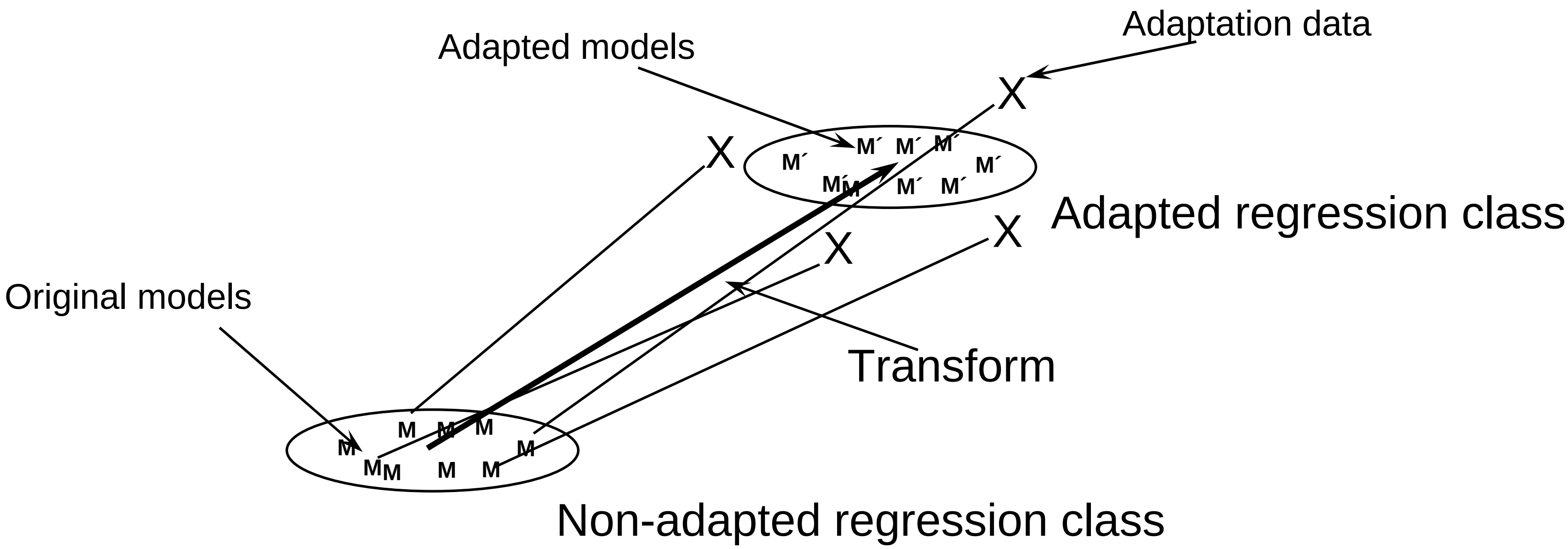
- Linear regression functions transform mean and covariance for maximizing the likelihood of the adaptation data

$$\bar{\boldsymbol{\mu}}_{ik} = \mathbf{A}_c \boldsymbol{\mu}_{ik} + \mathbf{b}_c$$

- $\mathbf{A}_c$ is a regression matrix, $\mathbf{b}_c$ is an additive vector for regression class c
- **A** and **b** can be estimated in a similar way as when training the continuous observation parameters
- Iteration for optimization
- Models not in the adaptation data are updated
- If little training data, use few regression classes
- Can adapt both means and variances
- Does not adapt transition probabilities

MLLR adaptation illustration

- The transform for a class is optimized to maximize the likelihood of the adapted models to generate the adaptation data



Speaker-Adaptive Training (SAT)

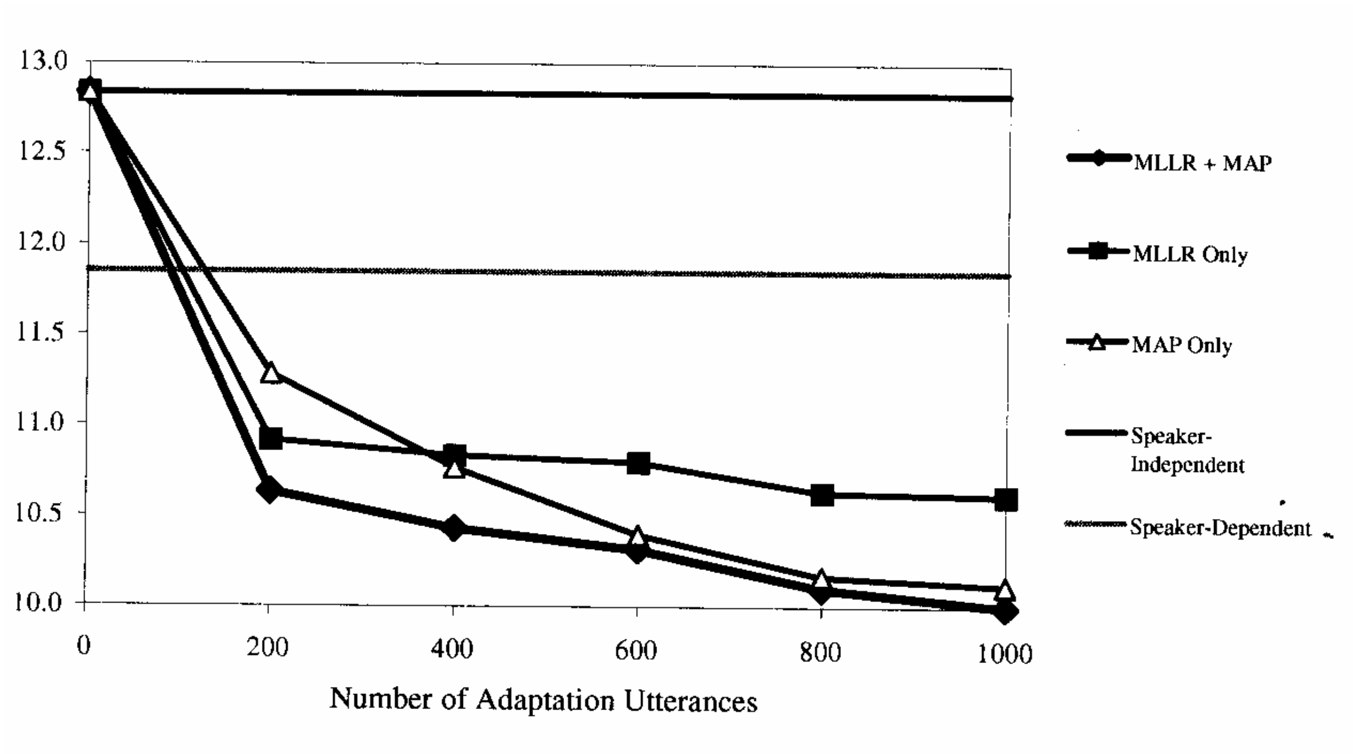
- Problem
  - Large variation in the speaker-independent models
    - MLLR adaptation of variances not very effective
- Solution: Speaker Adaptive Training
  - Adapt (MLLR) each training speaker to the speaker-independent model.
  - Use the adapted training data for each speaker to train a new speaker-independent model
  - Reduces the variation by moving all speakers towards their common average.
  - Requires adaptation during recognition

MLLR performance

Models	Relative Error Reduction
CHMM	Baseline
MLLR on mean only	+12%
MLLR on mean and variance	+2%
MLLR SAT	+8%

One context-independent regression class for all context-dependent phones with same mid unit

9.6.3 MLLR and MAP Comparison



- MLLR better for small adaptation data, MAP is better when the adaptation data is large. Combined MLLR+MAP best in both cases

9.6.4 Clustered Models

- A single speaker- and environment- independent model often has too much variability for high performance recognition and adaptation
- Cluster the training data into smaller partitions
- Gender-dependent models can reduce WER by 10%
- Speaker clustering can reduce it further, but not as much
- Environment clustering and adaptation in Chapter 10