Dies sind die in der Vorlesung

Sprachverstehen

Summer term 2016

Friedrich-Alexander-Universität Erlangen-Nürnberg

verwendeten Folien. Sie sind ausschließlich für den persönlichen Gebrauch zur Prüfungsvorbereitung bestimmt.

Weitere Quellen sind die empfohlenen Lehrbücher.

Erlangen, 18. Mai 2016
Elmar Nöth

Sprachverstehen

Summer term 2016

Elmar Nöth
Lehrstuhl für Informatik 5
(Mustererkennung)

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

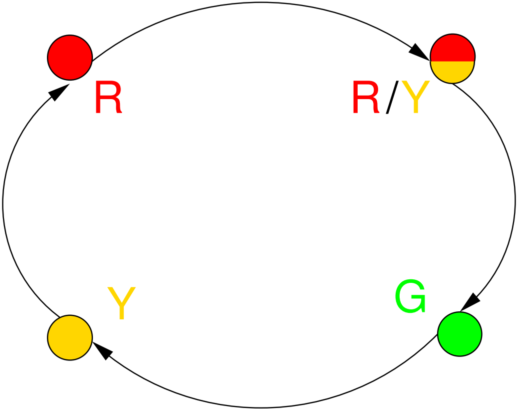
Teil V

Hidden Markov Models

An Observation

R RY G Y R RY G Y R RY G Y R RY G Y R RY G Y

An Automaton for the Observation



Properties of this Automaton?

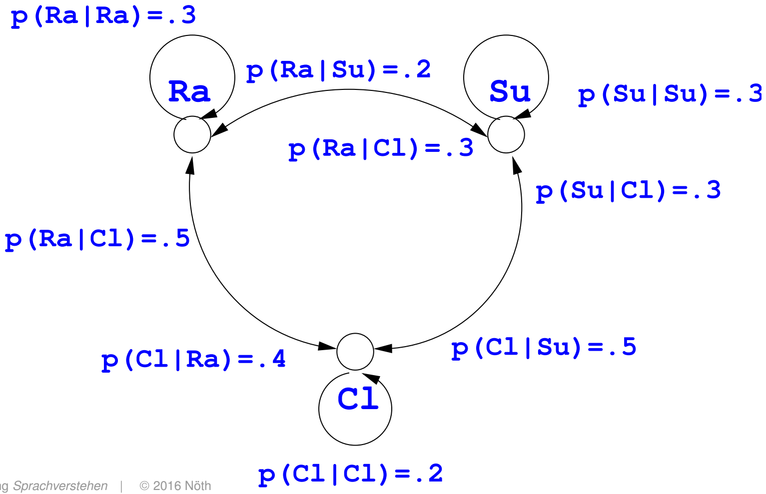
- Computer internal representation of the course of events of the real automaton
- Implementation as a directed graph
- Task: Automaton is supposed to learn the course of events from observing real traffic lights
- Here: Model ignores the time

A Weather Observation

RaRaRaSuSuSuSuClSuClClRaClSuRaRaRaSuSuRaRaSuSu
SuSuRaRaSuSuSuSuClRaRaRaSuSuSuSuweSuSuSuClSuClRa
...

- Behaviour is not deterministic any more
- Observation is made daily at 12h00 ⇒ equidistant, time factor

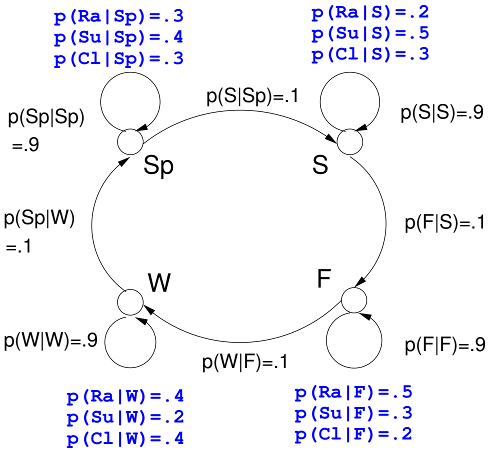
A Markov Chain for the Weather



Seasonal Variations

- Weather changes strongly depending on the season
- Season is not (exactly) observable
- Begin and end of the season is imprecise
- **SIMPLIFYING** first order **MARKOV** assumption:
  Probability for the weather **ONLY** depends on
  - The weather of the day before (observable) and
  - The season (not observable)

A Hidden Markov Model for the Weather



Isolated Word Recognition

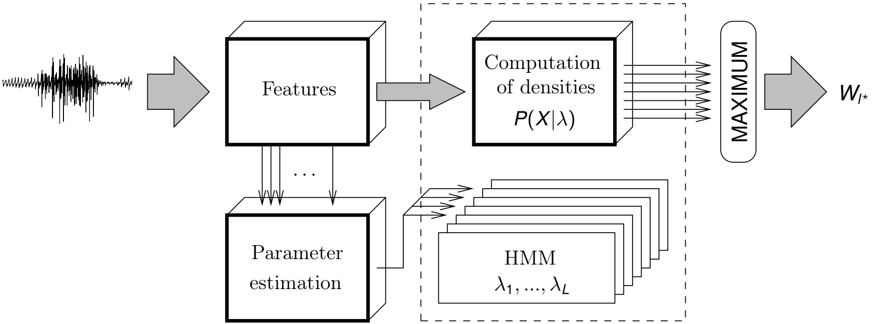
- Given: A set of words $\mathcal{W} = \{W_1, \ldots, W_L\}$
- Observed is a sequence of feature vectors $\boldsymbol{X} = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_T$ (utterance)
- Which word was uttered?

→ Decision according to the Bayes formula

$$l^* = \operatorname*{argmax}_l P(W_l \mid \boldsymbol{X}) \quad \text{with} \quad P(W_l \mid \boldsymbol{X}) = \frac{P(\boldsymbol{X} \mid W_l) \cdot P(W_l)}{P(\boldsymbol{X})}$$

- A priori probability of the words $P(W_l)$ is, e.g., estimated via counting a training sample
- $P(\boldsymbol{X} \mid W_l)$ can not approximated, e.g., with a Gaussian density function, because the length T of the utterance is variable
- however: $P(\boldsymbol{X} \mid W_l)$ can be estimated with Hidden Markov Models (HMMs)

Isolated Word Recognition



Hidden Markov Models

Task: Estimate the probability $P(x_1, x_2, .., x_T)$ for arbitrary values of T

- Problem 1: The stochastic process, which generates the x_t, is in reality not stationary, i.e ., $P(x_t)$ depends on the spoken phoneme therefore it is not possible to simply assume that $P(x_1, x_2, .., x_T) = \prod_{t=1}^{T} P(x_t)$
- Problem 2: Since the x_t can be feature vectors, to compute, e.g., $P(x_{t-1}, x_t)$ is very difficult
  if x_t is 24-dimensional, only 10 observations mean that a 240-dimensional density has to be estimated

Hidden Markov Models

Trick 1: Introduction of discrete, hidden states $q_t \in \{s_1, ..s_N\}$

- The x_t only depend on the current state q_t :
  $P(x_1, x_2, .., x_T) = \sum_{q_1,..,q_T} P(q_1) \cdot P(x_1|q_1) \cdot \prod_{t=2}^{T} P(x_t|q_t) \cdot P(q_t|q_1, ..q_{t-1})$

Trick 2: Assume that the stochastic process that generates the q_t is a stationary first order Markov process, i.e., the following holds:
$P(q_t \mid q_1, ..., q_{t-1}) = P(q_t \mid q_{t-1})$

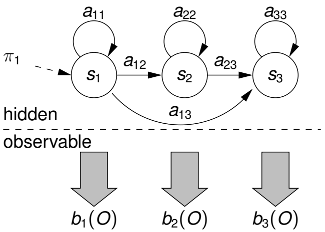
- then $P(x_1, x_2, .., x_T)$ can be computed as follows:
  $P(x_1, x_2, .., x_T) = \sum_{q_1,..,q_T} P(q_1) \cdot P(x_1|q_1) \cdot \prod_{t=2}^{T} P(x_t|q_t) \cdot P(q_t|q_{t-1})$

Hidden Markov Models

For the computation of $P(x_1, x_2, .., x_T)$ only the densities $P(x_t|q_t), q_t \in \{s_1, ..s_N\}$ and the probabilities $P(q_t|q_{t-1})$ are necessary

- Since the Markov process is stationary, the $P(q_t|q_{t-1})$ are independent of t and can be stored in an NxN matrix
- The $P(q_1)$ are N different values
- The $P(x_t|q_t)$ can be chosen depending on the application, e.g., N different Gaussian densities

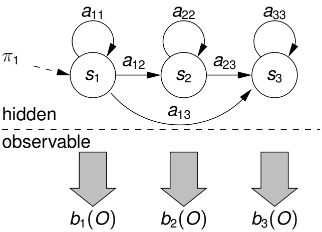
Hidden Markov Model: Definition



HMM $\boldsymbol{\lambda} = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B})$ with

- $\boldsymbol{\pi} = (\pi_i)$: $P(q_1)$ start probabilities
- $\boldsymbol{A} = [a_{ij}]$: $P(q_t = s_j | q_{t-1} = s_i)$ transition probabilities
- $\boldsymbol{B} = (b_j)$: $P(x_t | q_t = s_j)$ emission probabilities depending on model type:

Hidden Markov Model: Definition



- discrete HMM: finite emission alphabet, e.g., VQ codebook classes
- continuous HMM: Mixture densities
  $b_j(\boldsymbol{x}) = \sum_{k=1}^{K_j} \omega_{jk} \cdot \mathcal{N}(\boldsymbol{x}|\mu_{jk}, \Gamma_{jk})$
- semi-continuous HMM:
  $b_j(\boldsymbol{x}) = \sum_{k=1}^{K} \omega_{jk} \cdot \mathcal{N}(\boldsymbol{x}|\mu_{k}, \Gamma_{k})$

HMM: Four Problems

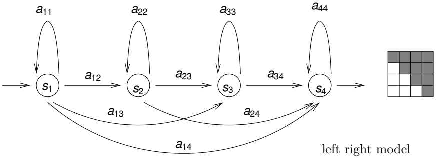
0. How to decide on the topology of the HMM, i.e. which transition probabilities $P(s_i|s_j) > 0$ will be allowed and which will be set to $P(s_i|s_j) = 0$?
1. How can the production probability $P(x_1, .., x_T|\lambda)$ be computed efficiently?
2. Decoding: What is the most likely transition sequence, given an observation $x_1, ..x_T$?
3. How can the parameters of the HMM be estimated from a training sample?

N.B.: The first problem is application dependent and a lot of hand crafting, so many people speak of three problems

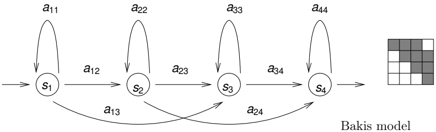
HMM Topologies (Problem 0)

- Topologies and number of states is often determined manually
- If all states are connected to each other, the model is called an ergodic HMM
- appropriate model topologies for speech recognition ($>$ 0 entries in the transition matrix are highlighted in grey)

HMM Topologies (Problem 0)

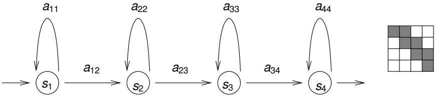


left right model



Bakis model

linear model



Estimation of the Production Probability (Problem 1)

Wanted: The probability $P(\boldsymbol{O} \mid \boldsymbol{\lambda})$ that $\boldsymbol{O} = O_1, ..O_T$ was generated by $\boldsymbol{\lambda}$

- As above, only in different writing: Computation of the production probability via summing up over all possible state sequences
  $P(\boldsymbol{O} \mid \boldsymbol{\lambda}) \;=\; \sum_{\boldsymbol{q} \in \mathcal{Q}^T} P(\boldsymbol{O}, \boldsymbol{q} \mid \boldsymbol{\lambda}) \;=\; \sum_{\boldsymbol{q} \in \mathcal{Q}^T} \pi_{q_1} b_{q_1}(O_1) \cdot \prod_{t=2}^{T} a_{q_{t-1} q_t} b_{q_t}(O_t)$
- About $2T \cdot N^T$ multiplications: Exponential complexity with T
- $\Rightarrow$ Polynomial complexity via Markov assumption
- Simplification by introducing the forward and backward probability α, β (only one probability necessary, but will need both later)
- Forward probability:
  $\alpha_t(j) = P(O_1 \ldots O_t, q_t = j \mid \boldsymbol{\lambda})$
- Backward probability:
  $\beta_t(i) = P(O_{t+1} \ldots O_T \mid q_t = i, \boldsymbol{\lambda})$
- For each time t it holds:
  $\alpha_t(j) \cdot \beta_t(j) \;=\; P(\boldsymbol{O}, q_t = j \mid \boldsymbol{\lambda})$ and $P(\boldsymbol{O} \mid \boldsymbol{\lambda}) \;=\; \sum_{j=1}^{N} \alpha_t(j)\beta_t(j)$

Estimation of the Production Probability (Problem 1)

2 equivalent algorithms, either with forward or backward probability

- **Initialisation**:
  For all $j = 1, .., N$

  $$\alpha_1(j) = \pi_j b_j(O_1)$$

  for all $i = 1, .., N$

  $$\beta_T(i) = 1$$

- **Recursion**: set
  for $t > 1$ and all $j = 1, .., N$

  $$\alpha_t(j) = \left(\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} \right) b_j(O_t)$$

  for $t < T$ and all $i = 1, .., N$

  $$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$$

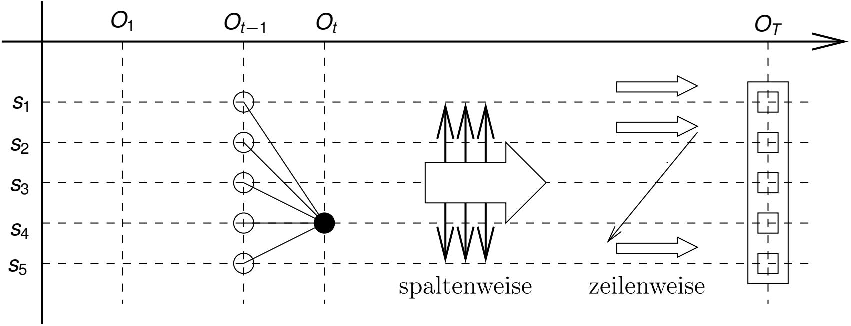
- **Termination**: compute

  $$P(\boldsymbol{O} \mid \boldsymbol{\lambda}) = \sum_{j=1}^{N} \alpha_T(j)$$

  $$P(\boldsymbol{O} \mid \boldsymbol{\lambda}) = \sum_{j=1}^{N} \pi_j b_j(O_1) \beta_1(j)$$

Estimation of the Production Probability (Problem 1)

- both algorithms need the same computing time
- The complexity is quadratic w.r.t. N and linear w.r.t. T: $2 \cdot N^2 \cdot T$ multiplications



Viterbi Algorithm (Problem 2)

We are looking for the state sequence $q_1, ..q_T$, which maximises $P(q_1, .., q_T \mid \boldsymbol{O}, \boldsymbol{\lambda})$, i.e., the most likely state sequence of the HMM during the observation

- **a posteriori probability for a state sequence $\boldsymbol{q}$**
  $P(\boldsymbol{q} \mid \boldsymbol{O}, \boldsymbol{\lambda}) \;=\; \frac{P(\boldsymbol{O},\boldsymbol{q}|\boldsymbol{\lambda})}{P(\boldsymbol{O}|\boldsymbol{\lambda})}$
- $\boldsymbol{q}^*$ is **optimal state sequence** if
  $P(\boldsymbol{O}, \boldsymbol{q}^* \mid \boldsymbol{\lambda}) \;=\; \max_{\boldsymbol{q} \in \mathcal{Q}^T} P(\boldsymbol{O}, \boldsymbol{q} \mid \boldsymbol{\lambda}) \;=:\; P^*(\boldsymbol{O} \mid \boldsymbol{\lambda})$
- **Viterbi Algorithm:** Alternative for the computation of the forward matrix
- The following probabilities are computed instead of the $\alpha_t(j)$:
  $\vartheta_t(j) \;=\; \max\{P(O_1 \ldots O_t, q_1 \ldots q_t \mid \boldsymbol{\lambda}) \mid \boldsymbol{q} \in \mathcal{Q}^T \text{ with } q_t = j\}$
- Back pointers for the extraction of the state sequence

Viterbi Algorithm (Problem 2)

- **Initialisation**:
  Set $\vartheta_1(j) = \pi_j b_j(O_1)$ and $\psi_1(j) = 0$. for all $j = 1, \ldots, N$
- **Recursion**: For all $j = 1, \ldots, N$ set

$$\vartheta_t(j) = \max_i(\vartheta_{t-1}(i)a_{ij})b_j(O_t) \quad \text{and} \quad \psi_t(j) = \operatorname*{argmax}_i \vartheta_{t-1}(i)a_{ij}$$

- **Termination**: Set

$$P^*(\boldsymbol{O} \mid \boldsymbol{\lambda}) = \max_j \vartheta_T(j) \quad \text{and} \quad q_T^* = \operatorname*{argmax}_j \vartheta_T(j)$$

- **Backtracking**: For $t = T - 1, \ldots, 1$ the optimal sequence results in

$$q_t^* = \psi_{t+1}(q_{t+1}^*)$$

ML Estimation of the Model Parameters (Problem 3)

Wanted: A set of HMM parameters $\hat{\boldsymbol{\lambda}}$, given a training utterance $\boldsymbol{O}$

- ML estimation: Choose $\hat{\boldsymbol{\lambda}}$ such that the following goal function is maximised:
  $\$_{\text{HMM}}(\boldsymbol{\lambda}) \;=\; \log P(\boldsymbol{O} \mid \boldsymbol{\lambda}) \;=\; \log \sum_{\boldsymbol{q} \in \mathcal{Q}^T} P(\boldsymbol{O}, \boldsymbol{q} \mid \boldsymbol{\lambda})$
- Observable random variable: $X = \boldsymbol{O}$
- Hidden random variable: $Y = \boldsymbol{q}$
- Parameter to be estimated: $B = \hat{\boldsymbol{\lambda}}$
- Connection between $\boldsymbol{O}$ and $\boldsymbol{q}$ is known
  $\rightarrow$ Application of the Expectation Maximisation (EM) algorithm
- Maximise $\hat{\boldsymbol{\lambda}}$ w.r.t. the **Kullback-Leibler Statistics**
  $Q(\boldsymbol{\lambda}, \hat{\boldsymbol{\lambda}}) \;=\; \sum_{\boldsymbol{q} \in \mathcal{Q}^T} P(\boldsymbol{q} \mid \boldsymbol{O}, \boldsymbol{\lambda}) \cdot \log P(\boldsymbol{O}, \boldsymbol{q} \mid \hat{\boldsymbol{\lambda}})$

ML Estimation of the Model Parameters (Problem 3)

For the computation of the a posteriori Probability $P(\boldsymbol{q} \mid \boldsymbol{O}, \boldsymbol{\lambda})$ the following variables are introduced:

- **a posteriori transition probability** for $s_i \rightarrow s_j$ at time t:

$$\begin{aligned} \xi_t(i,j) &= P(q_t = i, q_{t+1} = j \mid \boldsymbol{O}, \boldsymbol{\lambda}) \\ &= \frac{P(q_t = i, q_{t+1} = j, \boldsymbol{O} \mid \boldsymbol{\lambda})}{P(\boldsymbol{O} \mid \boldsymbol{\lambda})} \\ &= \frac{\alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \alpha_t(i)\beta_t(i)}, \quad 1 \leq t < T \end{aligned}$$

- **a posteriori state probability** for s_i at time t:
  $\gamma_t(i) = P(q_t = i \mid \boldsymbol{O}, \boldsymbol{\lambda}) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$
- Summing up $\xi_t(i,j)$ and $\gamma_t(i)$ across all t $\rightarrow$ Expected value for transitions $s_i \rightarrow s_j$ and stay in s_i, respectively

Baum-Welch Formulae (Problem 3)

The estimation formulae stemming from the EM algorithm for HMMs are referred to as Baum-Welch formulae, the training as Baum-Welch training oder Forward-backward algorithm

Baum-Welch Formulae for HMMs with discrete emission density:

$$\hat{\pi}_i = \gamma_1(i) = \frac{\alpha_1(i)\beta_1(i)}{\sum_{j=1}^{N} \alpha_1(j)\beta_1(j)}$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1})\beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i)\beta_t(i)}$$

$$\hat{b}_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j)\chi_{[O_t=v_k]}}{\sum_{t=1}^{T} \gamma_t(j)} = \frac{\sum_{t=1}^{T} \alpha_t(j)\beta_t(j)\chi_{[O_t=v_k]}}{\sum_{t=1}^{T} \alpha_t(j)\beta_t(j)}$$

Baum-Welch Formulae (Problem 3)

- $\chi_{[\cdot]} = 1$ for true statements and 0 otherwise
- Iterative application leads to a local optimum
- Effort of one iteration is only minimally higher than computation of the forward and backward matrix

Viterbi Training (Problem 3)

- Application of the EM* algorithm is referred to as Viterbi training
- Optimisation w.r.t. the Viterbi rating
  $P^*(\boldsymbol{O} \mid \boldsymbol{\lambda}) = P(\boldsymbol{O}, \boldsymbol{q}^* \mid \boldsymbol{\lambda}^{(n-1)})$
- **significantly more efficient** than BWT
- **less reliable** than BWT with small training samples
- VT corresponds to BWT with **modified a posteriori probabilities**
  $\gamma_t^*(i) \;=\; \chi_{[q_t^* = s_i]}$ and $\xi_t^*(i,j) \;=\; \chi_{[q_t^* = s_i, q_{t+1}^* = s_j]}$

Viterbi-Training (Problem 3)

- Choose a start model $\boldsymbol{\lambda}^{(0)}$
- For $n = 1, 2, \ldots$:
  (1) Search for the optimal state sequence $\boldsymbol{q}^*$ with

$$P(\boldsymbol{O}, \boldsymbol{q}^* \mid \boldsymbol{\lambda}^{(n-1)}) \;=\; \max_{\boldsymbol{q}} P(\boldsymbol{O}, \boldsymbol{q} \mid \boldsymbol{\lambda}^{(n-1)})$$

  using the Viterbi algorithm.

  (2) Compute the start, transition, and emission frequencies belonging to $\boldsymbol{q}^*$

$$\bar{\pi}_i \;=\; \chi_{[q_1 = s_i]}\,, \quad \bar{a}_{ij} \;=\; \sum_{t=1}^{T-1} \chi_{[q_t^* = s_i, q_{t+1}^* = s_j]} \quad \text{and} \quad \bar{b}_{jk} \;=\; \sum_{t=1}^{T} \chi_{[q_t^* = s_j, O_t = v_k]}$$

  (3) Normalise $\hat{\pi}_i = \bar{\pi}_i / \sum_i \bar{\pi}_i$, $\hat{a}_{ij} = \bar{a}_{ij} / \sum_j \bar{a}_{ij}$ and $\hat{b}_{jk} = \bar{b}_{jk} / \sum_k \bar{b}_{jk}$

  (4) Set $\boldsymbol{\lambda}^{(n)} \;=\; (\hat{\pi}_i, \hat{a}_{ij}, \hat{b}_{jk})$

Practical Use of HMMs

For long utterances the multiplication of many probabilities with limited numerical precision can quickly lead to values =0

- 1^{st} corrective: Introduce scaling/normalisation using time cycle dependent scaling factors C_t in the *forward algorithm*
  $\tilde{\alpha}_t(j) = \frac{1}{C_t} \cdot \alpha_t(j) = \frac{\alpha_t(j)}{\sum_i \alpha_t(i)}$
  $\alpha_t(j)$ can always be reconstructed, since $\alpha_t(j) = C_1 \cdot C_2 \cdots C_t \cdot \tilde{\alpha}_t(j)$
- 2^{nd} corrective: Take the logarithm:
  - In the Viterbi algorithm logarithmic probabilities are used $\rightarrow$ multiplication becomes addition
  - Problem: In the forward algorithm logarithmic probabilities have to be added $\rightarrow$ **Kingsbury-Rayner Formula**
    $\log(p_1 + p_2) = \log p_1 + \log(1 + e^{\log p_2 - \log p_1})$
  - Speed up with a table of $\log(1 + e^{\log p_2 - \log p_1})$

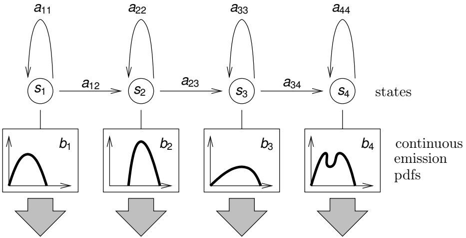
Practical Use of HMMs

- Initial estimates $\boldsymbol{\lambda}^{(0)}$ have to be found:
  - Use the LBG-algorithm to cluster all feature vectors and estimate an initial discrete HMM, which is the initialisation for the continuous HMM
  - The emission densities of the continuous HMMs can be initialised via clustering of the segmentation generated with the discrete HMMs
- if there are several training examples per HMM, the estimation formulae change just as with the EM algorithm
- e.g. the estimation formula for transition probability:

$$\hat{a}_{ij} = \frac{\sum_{l=1}^{L} \sum_{m=1}^{M_l} \left(\sum_{t=1}^{T_{l,m}-1} \xi_t^{(l,m)}(i,j) \right)}{\sum_{l=1}^{L} \sum_{m=1}^{M_l} \left(\sum_{t=1}^{T_{l,m}-1} \gamma_t^{(l,m)}(i) \right)}$$

Continuous HMM

- discrete HMMs need a previous VQ:
  $\boldsymbol{X} = \boldsymbol{x}_1 \ldots \boldsymbol{x}_T \longrightarrow \boldsymbol{O} = O_1, \ldots, O_T \Rightarrow$ **loss of information**
- **continuous emission densities** $b_j(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbf{R}^D$ to process the feature vectors: performance characterised via mixture densities $P(\boldsymbol{X}, \boldsymbol{q} \mid \boldsymbol{\lambda})$



- $b_j(\boldsymbol{x})$ from a **parametric family** of densities, e.g., Gaussian distribution densities $\mathcal{N}$

Continuous HMMs: Estimation Formulae

- Estimation formulae for start and transition probabilities and computation of the probabilities equivalent to discrete HMMs
- For Gaussian distributions:

$$b_j(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)$$

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

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

$$= \frac{1}{\sum_t \gamma_t(j)} \sum_{t=1}^{T} \gamma_t(j) \boldsymbol{x}_t \boldsymbol{x}_t^\top - \hat{\boldsymbol{\mu}}_j \hat{\boldsymbol{\mu}}_j^\top$$

Gaussian Mixture Densities

- Emission density is a Gaussian mixture density:

$$b_j(\boldsymbol{x}) = \sum_{k=1}^{K} c_{jk} g_{jk}(\boldsymbol{x}) = \sum_{k=1}^{K} c_{jk} \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_{jk}, \boldsymbol{\Sigma}_{jk}), \quad \sum_{k=1}^{K} c_{jk} = 1$$

- Any density function can be approximated with a large number of Gaussians
- Broadly used in Pattern Recognition
- The mixture component $k_t \in \mathcal{K}$ is a hidden variable as well
- Production probability

$$P(\boldsymbol{X} \mid \lambda) = \sum_{\boldsymbol{q} \in \mathcal{Q}^T} \sum_{\boldsymbol{k} \in \mathcal{K}^T} P(\boldsymbol{X}, \boldsymbol{q}, \boldsymbol{k} \mid \lambda)$$

Gaussian Mixture Densities: BW Estimation Formulae

a posteriori Selection probability of the components k in s_j at time t:

$$\zeta_t(j,k) = P(q_t = j, k_t = k \mid \boldsymbol{X}, \boldsymbol{\lambda}) = \begin{cases} \frac{1}{P(\boldsymbol{X}|\boldsymbol{\lambda})} \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} c_{jk} g_{jk}(\boldsymbol{x}_t) \beta_t(j) & \text{falls } t > 1 \\ \frac{1}{P(\boldsymbol{X}|\boldsymbol{\lambda})} \sum_{i=1}^{N} \pi_j c_{jk} g_{jk}(\boldsymbol{x}_1) \beta_1(j) & \text{falls } t = 1 \end{cases}$$

Estimation formulae:

$$\hat{c}_{jk} = \frac{1}{\sum_t \gamma_t(j)} \sum_{t=1}^{T} \zeta_t(j,k)$$

$$\hat{\boldsymbol{\mu}}_{jk} = \frac{1}{\sum_t \zeta_t(j,k)} \sum_{t=1}^{T} \zeta_t(j,k) \boldsymbol{x}_t$$

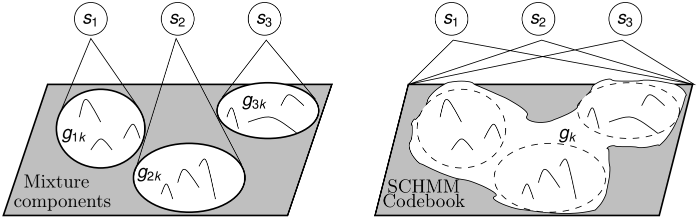
$$\hat{\boldsymbol{\Sigma}}_{jk} = \frac{1}{\sum_t \zeta_t(j,k)} \sum_{t=1}^{T} \zeta_t(j,k) \boldsymbol{x}_t \boldsymbol{x}_t^\top - \hat{\boldsymbol{\mu}}_{jk} \hat{\boldsymbol{\mu}}_{jk}^\top$$

Semi-continuous HMMs (SCHMM)

- Markov models with **semi-continuous** emission densities:

$$b_j(\boldsymbol{x}) = \sum_{k=1}^{K} c_{jk} g_k(\boldsymbol{x}) = \sum_{k=1}^{K} c_{jk} \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) , \quad \sum_{k=1}^{K} c_{jk} = 1$$

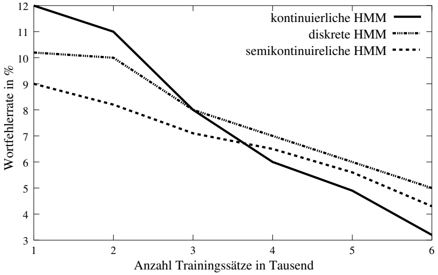
- Difference to continuous HMMs with mixture densities: Components for **one** state, now for **all** states
- SCHMM has ability to approximate mixture densities but needs less parameters



Semi-continuous HMM

- Mixture weights c_{ij} can also be considered to be emission probabilities of a discrete HMM
- SCHMM evaluates **all** codebook classes, density values $g_k(\boldsymbol{x})$ weigh the discrete emission probability (**soft** vector quantisation)
- VQ is part of the SCHMM
- the SCHMM can also be placed between discrete and continuous HMM

Properties of Semi-continuous HMMs



- Properties of a SCHMM:
  - Compact parameter space
  - No distortion due to quantisation
  - Inclusion of the VQ in the process of model optimisation
- With little training data the SCHMM is better than a continuous HMM, with much more data it is worse

Semi-continuous HMM: BW Estimation Formulae

Summation of the density statistics over all states

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{\sum_t \sum_j \zeta_t(j,k)} \sum_{t=1}^{T} \sum_{j=1}^{N} \zeta_t(j,k) \boldsymbol{x}_t$$

$$\hat{\boldsymbol{\Sigma}}_k = \frac{1}{\sum_t \sum_j \zeta_t(j,k)} \sum_{t=1}^{T} \sum_{j=1}^{N} \zeta_t(j,k) \boldsymbol{x}_t \boldsymbol{x}_t^\top - \hat{\boldsymbol{\mu}}_k \hat{\boldsymbol{\mu}}_k^\top$$