Hidden Markov Models – an Introduction

Jan Černocký
Brno University of Technology, Faculty of Information Technology

Contents

1 Introduction

- HMMs are the representative of the statistical approach to speech recognition.
- HMMs, as well as DTW (Dynamic Time Warping), have to take into account:
  - the variability of speech in the parametric space
  - the temporal variability

Comparison of DTW and HMM:

	DTW	**HMM**
time	different paths with different lengths	different state sequences (the states can be skipped or repeated)
parameters	the variability is evaluated using distances	probability density functions (pdf) are used, not "hard" values of vectors

2 Recognition of isolated words

We have:

- an input sequence (parameter matrix)

$$\mathbf{O} = [\mathbf{o}(1), \mathbf{o}(2), \ldots, \mathbf{o}(T)], \tag{1}$$

  where $\mathbf{o}(1), \mathbf{o}(2), \ldots, \mathbf{o}(T)$ are parameter vectors computed for each frame of the input signal
- a dictionary of $\check{N}$ words w_i, $i \in [1, \check{N}]$.

We want to know, to which word belongs the input sequence.

2.1 Probabilistic formulation of the problem

$$i^\star = \arg\max_i \{\mathcal{P}(w_i|\mathbf{O})\}, \tag{2}$$

where $\mathcal{P}(w_i|\mathbf{O})$ is the conditional probability of the word w_i knowing $\mathbf{O}$.

Illustration of conditional probabilities:

$$\mathcal{P}(problem|\check{S}koda) >> \mathcal{P}(problem|Mercedes). \tag{3}$$

In the speech recognition, we are unfortunately not able to evaluate $\mathcal{P}(w_i|\mathbf{O})$ directly. The formula of Bays helps us (we are going to omit the index i):

$$\mathcal{P}(w|\mathbf{O}) = \frac{\mathcal{P}(\mathbf{O}|w)\mathcal{P}(w)}{\mathcal{P}(\mathbf{O})}, \tag{4}$$

where

- $\mathcal{P}(w)$ is the a-priori (known) probability of words, which is determined by the language model (LM). In case the words have equal probabilities, it will not affect the maximization.
- $\mathcal{P}(\mathbf{O})$ is the a-priori probability of the observation sequence. We can not evaluate this probability, but as it is constant for all the words, it will not affect the maximization.
- $\mathcal{P}(\mathbf{O}|w)$ is the most important probability: the probability of the sequence $\mathbf{O}$ knowing the word w. To compute it, we should have a *model* M for the word w. We imagine, that the model *generates* the sequence $\mathbf{O}$.

The model is a finite automaton, with several states. Between the states, we find the transition probabilities a_{ij}. Except the first and the last state, all states are *emitting* (we can imagine, that they could generate some vectors). Each of those emitting states has attached an *emission probability density function (pdf)* $b_j(\mathbf{o})$.

A possible configuration of a model is given in Figure 1.

2.2 Transition probabilities a_{ij}

should obey:

$$\sum_j a_{ij} = 1 \tag{5}$$

Our example: only three types of transition probabilities:

- $a_{i,i}$ probability of staying in a state.
- $a_{i,i+1}$ probability of going to the following state.
- $a_{i,i+2}$ probability of skipping the following state.

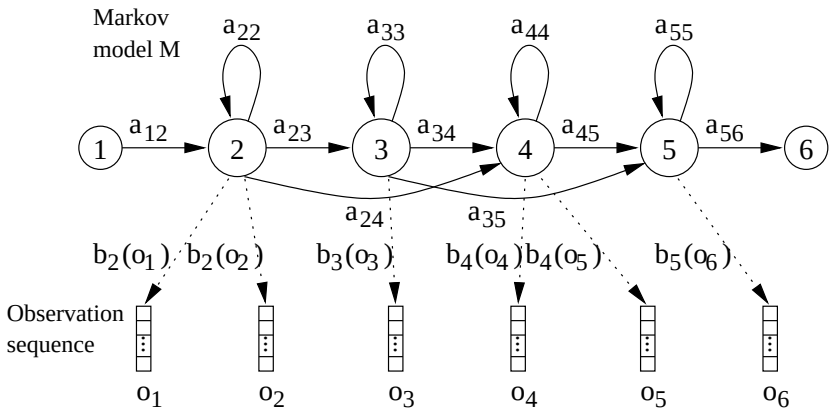

Figure 1: Example configuration of an HMM and association of input vectors with states. The corresponding state sequence is $X = [1, 2, 2, 3, 4, 4, 5, 6]$.

We can write a matrix of transition probabilities:

$$\mathbf{A} = \begin{bmatrix} 0 & a_{12} & 0 & 0 & 0 & 0 \\ 0 & a_{22} & a_{23} & a_{24} & 0 & 0 \\ 0 & 0 & a_{33} & a_{34} & a_{35} & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} & 0 \\ 0 & 0 & 0 & 0 & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6}$$

The majority of models used in ASP (automatic speech processing) are left–right (so that from a state, we can not return to preceding ones): $a_{i,j<i} = 0$.

2.3 Emission probability density functions (pdfs)

We note the probability of emission of vector $\mathbf{o}(t)$ by i-th state $b_i[\mathbf{o}(t)]$. There are two possibilities of defining those pdfs: discrete pdfs, and continuous pdfs.

2.3.1 Discrete pdfs

In this case, the P-dimensional observation vectors are first converted to *discrete symbols* using a classifier based on minimum distance (vector quantizer). An example for two element vectors $\mathbf{o}(t)$ is given in Figure 2. s_1 till s_9 are symbols, the codebook size in this case is L=9. The emission pdfs are given by tables of probabilities. For each emitting state j, we can define:

$b_j(s_1)$
$b_j(s_2)$
...
$b_j(s_L)$

Those models are called **discrete HMMs**.

2.3.2 Continuous pdfs

Here, the probability density functions are given using distributions or their sums. Distributions are determined by their **parameters**. Most often, the Gaussian distribution is used.

If the vector $\mathbf{o}$ had only one element (in reality, $\mathbf{o}$ has never one element), the density for j-th state would be given by one-dimensional distribution:

$$b_j[o(t)] = \mathcal{N}(o(t); \mu_j, \sigma_j) = \frac{1}{\sigma_j\sqrt{2\pi}} e^{-\frac{[o(t)-\mu_j]^2}{2\sigma_j^2}} \tag{7}$$

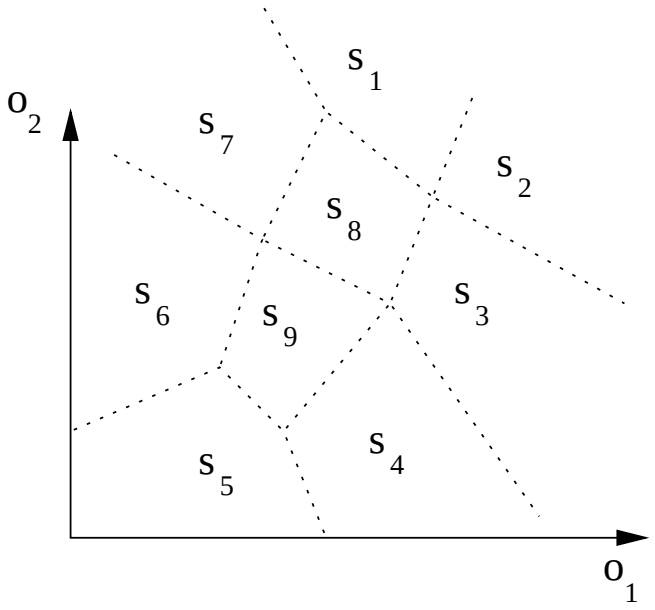

Figure 2: Conversion of 2-element vectors **o** to symbols using VQ.

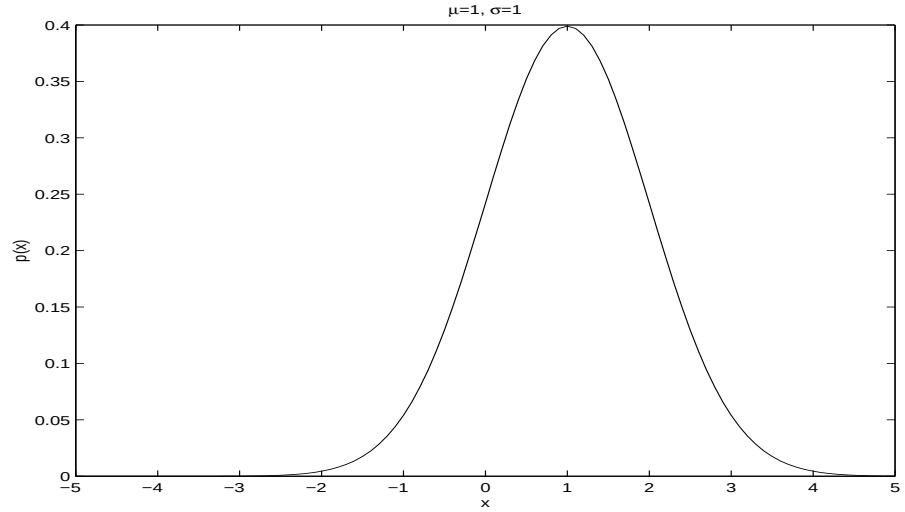

Figure 3: One-dimensional Gaussian distribution.

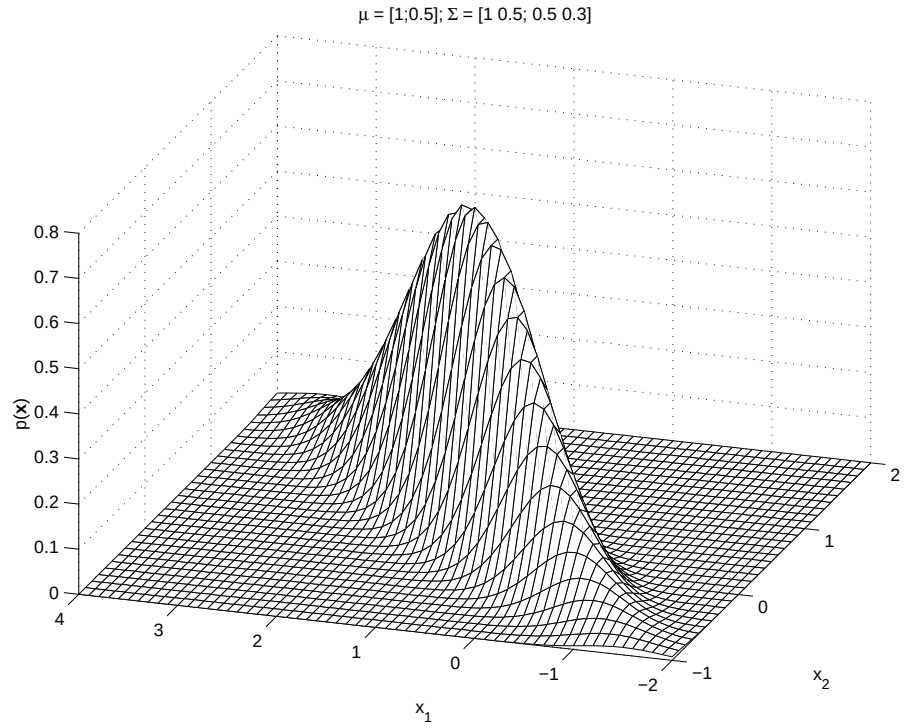

Figure 4: Two-dimensional Gaussian distribution.

where $\mathcal{N}$ denotes the normal distribution, μ_j is a scalar mean value, and σ_j is a scalar standard deviation (std). An example of such Gaussian distribution is given in Figure 3. In reality, we have multi-dimensional parameter vectors (dimension P), so that the distribution will be multi-dimensional as well:

$$b_j[\mathbf{o}(t)] = \mathcal{N}(\mathbf{o}(t); \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) = \frac{1}{\sqrt{(2\pi)^P |\boldsymbol{\Sigma}_j|}} e^{-\frac{1}{2}(\mathbf{o}(t)-\boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\mathbf{o}(t)-\boldsymbol{\mu}_j)}, \tag{8}$$

where $\boldsymbol{\mu}_j$ is the vector mean, $\boldsymbol{\Sigma}_j$ is the covariance matrix and $|\boldsymbol{\Sigma}_j|$ denotes the determinant. An example of such distribution with P=2 is given in Figure 4.

In most recognition systems however, we are trying to make the parameters within the vector uncorrelated (or at least we believe that they are uncorrelated), so that the covariance matrix is diagonal. It is then enough, instead of $P \times P$ covariance coefficients, to estimate P standard deviations which form the vector $\boldsymbol{\sigma}_j$. As consequence, the P-dimensional distribution will be a product of P independent 1-dimensional distributions:

$$b_j[o(t)] = \prod_{i=1}^{P} \mathcal{N}(o(t); \mu_{ji}, \sigma_{ji}) = \prod_{i=1}^{P} \frac{1}{\sigma_{ji}\sqrt{2\pi}} e^{-\frac{[o(t)-\mu_{ji}]^2}{2\sigma_{ji}^2}} \tag{9}$$

and the models will have less parameters ($2P$ per emitting state instead of $P + P \times P$), which can be more reliably estimated.

In speaker-independent systems however, modeling the pdf with only one Gaussian is often not enough to capture characteristics of all speakers. More complicated distributions are used:

1. Gaussian distribution mixtures (more P-dimensional distributions, summed up with weights). An example of such Gaussian mixture with 2 mixture components is given in Figure 5.
2. the parameter vectors can be divided into streams, for example:
   - **1st stream:** MFCC coefficients.
   - **2nd stream:** ΔMFCC coefficients – "velocities"
   - **3rd stream:** $\Delta\Delta$MFCC coefficients – "accelerations"

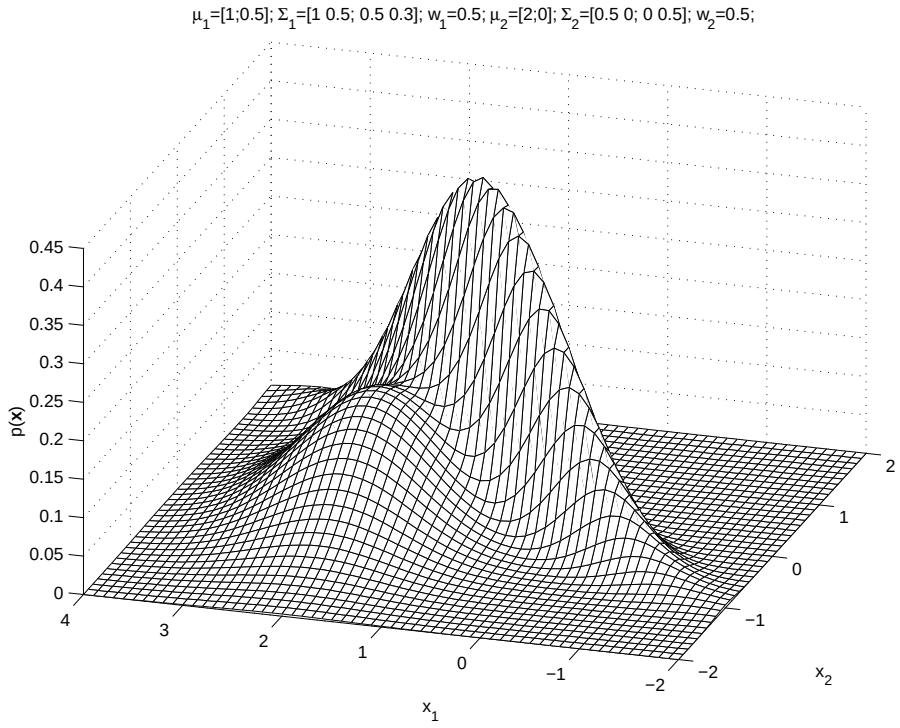

Figure 5: Two-dimensional Gaussian mixture with 2 mixture components.

- **4th stream:** E, ΔE and $\Delta\Delta$E – log energy, its velocity and acceleration.

and each stream has its own pdf (1 Gaussian or Gaussian mixture).

Steps 1) can result in very complex models with big number of parameters (several millions). To limit the number of parameters, some sets of parameters can be shared (tied) across states and/or models. Consequence: less parameters, more reliable estimation.

Remark: the mathematicians would say that the output value of a pdf, given a vector **o***,* **is not a probability**. *For simplicity reasons, we are going to call it* **a probability**...

2.4 State sequences

The observation vectors can be spread across the states using *state sequence* X of length $T+2$ (T being the length of observation sequence $\mathbf{O}$), indexed from $t=0$ to $t=T+1$. We can call X a *path through the model* (similarly as for DTW). In our example:

$$X = [1, 2, 2, 3, 4, 4, 5, 6] \tag{10}$$

Each state sequence must contain the 1st state (No. 1) in non-existing time $t=0$ and the last state (No. 6 in our example) in non-existing time $t=T+1$, even though no vector belongs to them (no vector can belong to them, as they are not emitting !). See Figure 6.

In the majority of cases, the state sequence is not "visible" from the exterior. This is why the models are called *hidden*.

3 Determination of probability, that the model M generates sequence O

first, we define a probability of generation of $\mathbf{O}$ on the path X:

$$\mathcal{P}(\mathbf{O}, X|M) = a_{x(o)x(1)} \prod_{t=1}^{T} b_{x(t)}(\mathbf{o}_t) a_{x(t)x(t+1)}, \tag{11}$$

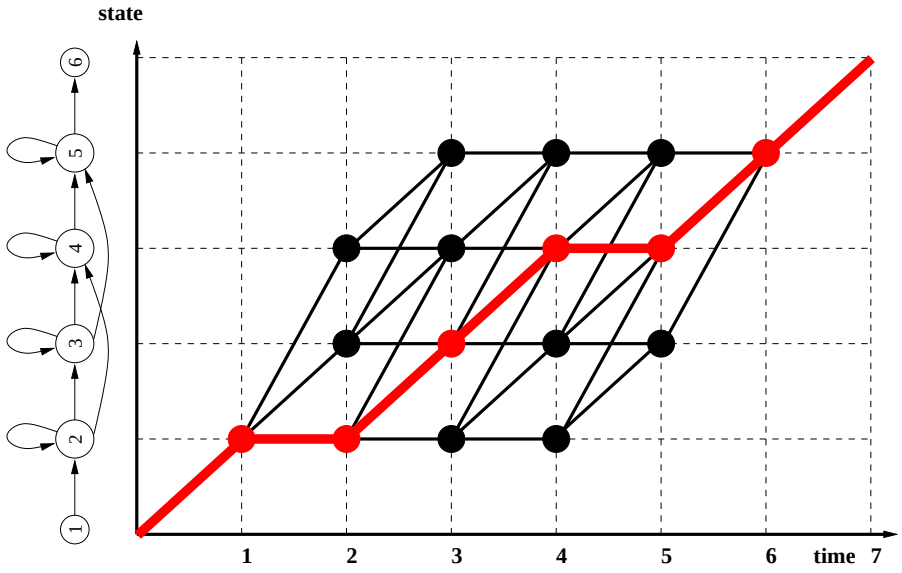

Figure 6: Illustration of possible state sequences. The dots stand for emissions of vectors by states. States 1 and 6 of the the HMM are non-emitting. The sequence $X = [1, 2, 2, 3, 4, 4, 5, 6]$ used as example is printed in red.

(it is actually a product of all possible probabilities, we encounter on our way through the model). In our example:

$$\mathcal{P}(\mathbf{O}, X|M) = a_{12}b_2(\mathbf{o}_1)a_{22}b_2(\mathbf{o}_2)a_{23}b_3(\mathbf{o}_3)\dots \tag{12}$$

There are two possibilities to define a unique probability, that the model generates the observation sequence:

a)

$$\mathcal{P}(\mathbf{O}|M) = \sum_{\{X\}} \mathcal{P}(\mathbf{O}, X|M), \tag{13}$$

where we take the sum of *all possible paths* of length $T + 2$ through the model. This is called Baum-Welch probability.

b)

$$\mathcal{P}^\star(\mathbf{O}|M) = \max_{\{X\}} \mathcal{P}(\mathbf{O}, X|M), \tag{14}$$

where we take only the maximum probability (that of the "best" path). We call it Viterbi probability.

Remarks

1. In case of DTW, we have *minimized the distance*. Here, we *maximize the probability*, sometimes also called likelihood and denoted $\mathcal{L}$.
2. For the computation of both Baum-Welch and Viterbi probabilities, fast algorithms are known: it is not necessary to evaluate the probabilities over all possible paths X.

4 Parameter training – the simple case

Till know, we expected that the parameters of models were known. Unfortunately usually nobody gives them to us. What we will be given is a set of observation sequences (e.g. parameter matrices) with transcriptions. We will be able to *train the parameters* of models M_i for word w_i from training sequences carrying the label w_i. Usually, we need many examples of a word to train a single model. Here, we will show the training on one single example.

The training proceeds in two steps:

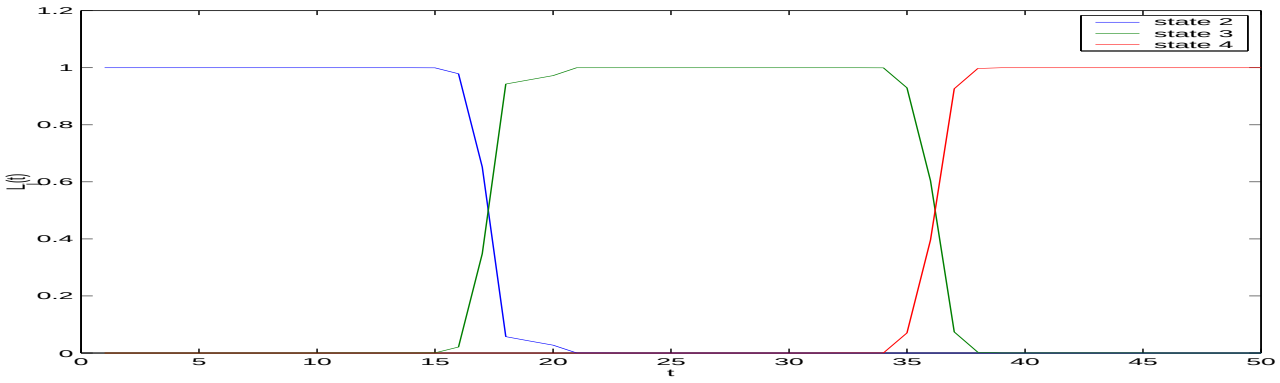

Figure 7: Example of state occupation functions for 3 emitting states of a model

1. the parameters of model are roughly estimated: for example, we take the global mean $\boldsymbol{\mu}$, and global covariance matrix $\boldsymbol{\Sigma}$ (or global standard deviation vector $\boldsymbol{\sigma}$) and we set the parameters of each state equal to those values: $\boldsymbol{\mu}_j = \boldsymbol{\mu}$ and $\boldsymbol{\Sigma}_j = \boldsymbol{\Sigma}$ (or $\boldsymbol{\sigma}_j = \boldsymbol{\sigma}$):

$$\begin{aligned} \hat{\boldsymbol{\mu}} &= \frac{1}{T}\sum_{t=1}^{T}\mathbf{o}(t) \\ \hat{\boldsymbol{\Sigma}} &= \frac{1}{T}\sum_{t=1}^{T}(\mathbf{o}(t)-\boldsymbol{\mu})(\mathbf{o}(t)-\boldsymbol{\mu})^T \quad \text{full covariance matrix} \\ \hat{\boldsymbol{\sigma}} &= \frac{1}{T}\sum_{t=1}^{T}(\mathbf{o}(t)-\boldsymbol{\mu})^2 \quad \text{diagonal covariance matrix} \end{aligned} \tag{15}$$

   It is also possible to allocate the vectors uniformly to states, and evaluate $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ (or $\boldsymbol{\sigma}_j$) for each state.

2. Iterations:

   a) Assignment of vectors to states. This assignment can be done "hardly", but more often, a "soft" assignment is computed. The state occupation function $L_j(t)$ determines, how much a vector will contribute to re-estimation of given state. An example of $L_j(t)$ is in Fig. 7.

   b) Re-estimation of parameters based on above computed assignment. In case the state occupation functions are used:

$$\begin{aligned} \hat{\boldsymbol{\mu}}_j &= \frac{\sum_{t=1}^{T} L_j(t)\mathbf{o}(t)}{\sum_{t=1}^{T} L_j(t)} \\ \hat{\boldsymbol{\Sigma}}_j &= \frac{\sum_{t=1}^{T} L_j(t)(\mathbf{o}(t)-\boldsymbol{\mu}_j)(\mathbf{o}(t)-\boldsymbol{\mu}_j)^T}{\sum_{t=1}^{T} L_j(t)} \quad \text{full covariance matrix} \\ \hat{\boldsymbol{\sigma}}_j &= \frac{\sum_{t=1}^{T} L_j(t)(\mathbf{o}(t)-\boldsymbol{\mu}_j)^2}{\sum_{t=1}^{T} L_j(t)} \quad \text{diagonal covariance matrix} \end{aligned} \tag{16}$$

   Similarly, we can derive also formulas for the computation of transition probabilities a_{ij} from state occupation functions (see [3]).

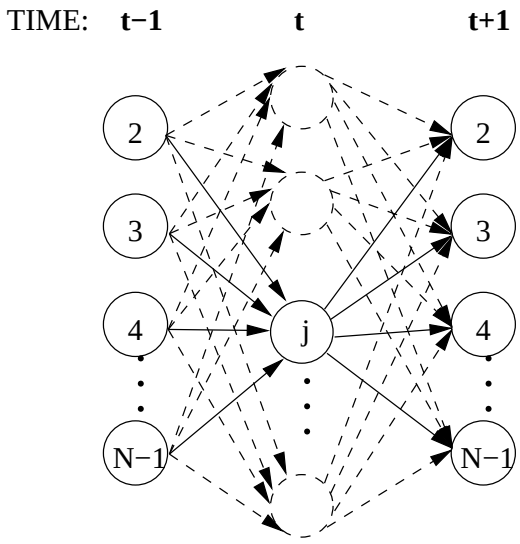

Figure 8: Paths crossing state j at time t (solid lines) and other states at time t (dashed lines).

Stop criterion: fixed number of iterations, or the probabilities evaluated during the iterations stop to change.

4.1 Computation of state occupation functions — FW-BW algorithm

The state occupation function $L_j(t)$ is the probability of being in state j at time t. It can be computed as the sum of probabilities of all the paths, that "cross" state j at time t (solid lines in Figure 8): $P(\mathbf{O}, x(t) = j|M)$. We want to be sure, however, that:

$$\sum_{j=1}^{N} L_j(t) = 1, \tag{17}$$

in other words: all states' "claims" to one vector must sum up to 100%. Therefore, we will normalize the probability of being in state j at time t by the probability of being in all states at time j (dashed *and* solid lines in Fig. 8):

$$L_j(t) = \frac{P(\mathbf{O}, x(t) = j|M)}{\sum_j P(\mathbf{O}, x(t) = j|M)}. \tag{18}$$

Being in all states at time t (knowing that we could be anywhere before and we can be anywhere after the time t) however means that we can take any path through the model. And sum of probabilities of all the paths is the Baum-Welch probability. We can therefore rewrite equation 18 to more common form:

$$L_j(t) = \frac{P(\mathbf{O}, x(t) = j|M)}{P(\mathbf{O}|M)}. \tag{19}$$

To determine probability of being in state j at time t: $P(\mathbf{O}, x(t) = j|M)$, it is very practical to define a probability of all paths ending in state j at time t and a probability of all paths beginning in state j at time t and extending toward the end of time. We will call them *partial forward probability* and *partial backward probability*.

4.1.1 Partial forward probability

For the model M with N states, the partial forward probability of being in state j at time t is:

$$\alpha_j(t) = \mathcal{P}\left(\mathbf{o}(1) \dots \mathbf{o}(t), x(t) = j|M\right) \tag{20}$$

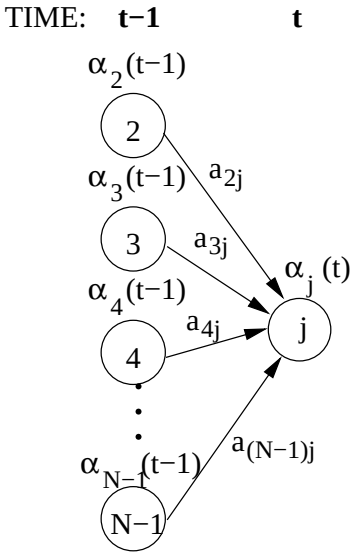

Figure 9: Computation of partial forward probability $\alpha_j(t)$

We find, that this forward probability can be evaluated using a recursive formula, if we take into account, that the state j can be reached only from other emitting states (Fig. 9):

$$\alpha_j(t) = \left[\sum_{i=2}^{N-1} \alpha_i(t-1)a_{ij}\right] b_j[\mathbf{o}(t)] \quad \text{for} \quad 2 \leq j \leq N-1 \tag{21}$$

The algorithm is initialized for time $t = 1$ by considering the transition probability from the 1st non-emitting state to all emitting states:

$$\alpha_j(1) = a_{1j}b_j[\mathbf{o}(1)] \quad \text{for} \quad 2 \leq j \leq N-1 \tag{22}$$

Once all $\alpha_j(1)$ are initialized, we can proceed in steps from $t = 1$ till $t = T$ and compute iteratively $\alpha_j(t)$ using Equation 21 at each step. When we are at the end ($t = T$), we will complete the computation by the evaluation of $\alpha_N(T+1)$ for the last non-emitting state (the time $T+1$ is only formal, as we know, that no vector, and hence no time, belongs to the N-th state):

$$\alpha_N(T+1) = \sum_{i=2}^{N-1} \alpha_i(T)a_{iN} \tag{23}$$

or in words "a sum of last alphas multiplied by transition probabilities to the last state". The last alpha is equal to the Baum-Welch probability:

$$P(\mathbf{O}|M) = \alpha_N(T+1) \tag{24}$$

so that we dispose of the way to compute the probability of emission of $\mathbf{O}$ by the model: $P(\mathbf{O}|M)$.

4.1.2 Partial backward probability

For the model M with N states, the partial backward probability of being in state j at time t is:

$$\beta_j(t) = \mathcal{P}\left(\mathbf{o}(t+1)\ldots\mathbf{o}(T)|x(t) = j, M\right) \tag{25}$$

We find, that this forward probability can be evaluated using a recursive formula, having in mind, that the state j can be reached only from other emitting states (Fig. 10). In this case, we do not take into account the emission probability for the time t, as it was the case in the computation of the forward probability, but for the time $t+1$:

$$\beta_j(t) = \sum_{i=2}^{N-1} a_{ji}b_i[\mathbf{o}(t+1)]\beta_i(t+1) \quad \text{for} \quad 2 \leq j \leq N-1 \tag{26}$$

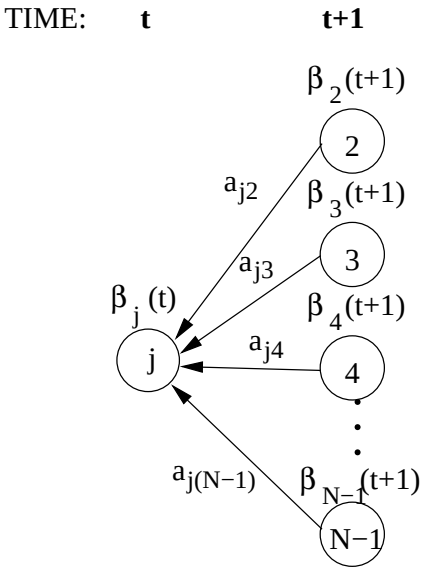

Figure 10: Computation of partial backward probability $\beta_j(t)$

We should initialize the algorithm at the last time T by the transition probabilities to the last (non-emitting) state:

$$\beta_j(T) = a_{jN} \quad \text{for} \quad 2 \leq j \leq N-1 \tag{27}$$

Once all $\beta_j(T)$ are initialized, we can proceed in steps against the time from $t = T$ till $t = 1$ and compute iteratively $\beta_j(t)$ using Equation 26 at each step. When we are at the beginning ($t = 1$), we will complete the computation by the evaluation of $\beta_1(0)$ for the first non-emitting state (the time 0 is only formal, as we know, that no vector, and hence no time, belongs to the 1st state):

$$\beta_1(0) = \sum_{i=2}^{N-1} a_{1i} b_i[\mathbf{o}(1)] \beta_i(1) \tag{28}$$

The first beta (first beta is the last computed one) is again equal to the Baum-Welch probability:

$$P(\mathbf{O}|M) = \beta_1(0) \tag{29}$$

so that we dispose of a second way to evaluate the probability of emission of $\mathbf{O}$ by the model.

4.1.3 Why did we do it all ?

¿From $\alpha_j(t)$ and $\beta_j(t)$, we can compute the probability of being in state j at the time t:

$$P(\mathbf{O}, x(t) = j|M) = \alpha_j(t)\beta_j(t) \tag{30}$$

The state occupation function $L_j(t)$ can be derived from this expression by a normalization by the total emission probability (see Eq. 19):

$$L_j(t) = \frac{\alpha_j(t)\beta_j(t)}{P(\mathbf{O}|M)} \tag{31}$$

where $P(\mathbf{O}|M)$ is given using Equations 29 or 24.

4.2 Algorithm of model training

Having all the formalism in place, the algorithm for model training can look like:

1. Allocate an *accumulator* for each estimated vector/matrix.

2. Compute the forward and backward probabilities $\alpha_j(t)$ and $\beta_j(t)$. Compute the values of state occupation functions $L_j(t)$.
3. To each accumulator, add the contribution of the vector $\mathbf{o}(t)$ weighted by the respective $L_j(t)$.
4. Use the final value of the accumulator for the computation of the estimated vector/matrix (here, we must divide by the sum $\sum_{t=1}^{T} L_j(t)$ – a standard normalization).
5. If the value $\mathcal{P}(\mathbf{O}|M)$ did not change significantly from the last iteration, **stop**. Otherwise, return to Step 1.

A great advantage of the above algorithm is the possibility to use arbitrary number of training sequences in steps 2 and 3. Above, this is not reflected – all the equations should be modified by adding sums and normalizations over all the utterances. See [3] for complete re-estimation formulae.

5 Recognition using the Viterbi decoding

Remainder: We should recognize an unknown sequence $\mathbf{O}$. We dispose of a dictionary of $\check{N}$ words $w_1 \dots w_{\check{N}}$. Each of them is modeled by a Hidden Markov model: $M_1 \dots M_{\check{N}}$. We want to know, which model would generate $\mathbf{O}$ with the highest probability:

$$i^\star = \arg\max_i \{\mathcal{P}(\mathbf{O}|M_i)\} \tag{32}$$

(we assume that all words have equal probabilities, so that the a-priori probability $\mathcal{P}(w_i)$ will not play any role...) For this evaluation, we could use the Baum-Welch probability

$$P(\mathbf{O}|M) = \alpha_N(T) = \beta_1(1) \tag{33}$$

but more likely, we will use the Viterbi probability for the most probable sequence of states, which can be computed more efficiently:

$$\mathcal{P}^\star(\mathbf{O}|M) = \max_{\{X\}} \mathcal{P}(\mathbf{O}, X|M). \tag{34}$$

and equation 32 will be rewritten:

$$i^\star = \arg\max_i \{\mathcal{P}^\star(\mathbf{O}|M_i)\}. \tag{35}$$

For the evaluation of the Viterbi probability, a similar algorithm as for the Baum-Welch one is used, but the operator "max" replaces the sums in the equations.

The *partial* Viterbi *probability* is defined:

$$\Phi_j(t) = \mathcal{P}^\star(\mathbf{o}(1) \dots \mathbf{o}(t), x(t) = j|M). \tag{36}$$

For j-th state and the time t, it can be evaluated in the similar way as the partial forward probability $\alpha_j(t)$:

$$\Phi_j(t) = \max_i \{\Phi_i(t-1)a_{ij}\} b_j[\mathbf{o}(t)] \quad \text{for} \quad 2 \le j \le N-1 \tag{37}$$

The computation differs from Eq. 21 only by replacing a sum with a maximum selection (illustrated in Fig. 11) Once again, the Φ for the first non-emitting state must be initialized as:

$$\Phi_j(1) = a_{1j} b_j[\mathbf{o}(1)] \quad \text{for} \quad 2 \le j \le N-1. \tag{38}$$

Once $\Phi_j(1)$ are initialized, we can proceed to computation of $\Phi_j(t)$ using Equation 37. At the end, we evaluate the partial probability for the last state:

$$\Phi_N(T+1) = \max_i \{\Phi_i(T)a_{iN}\} \tag{39}$$

which is actually the desired final Viterbi probability:

$$\mathcal{P}^\star(\mathbf{O}|M) = \Phi_N(T+1) \tag{40}$$

If we remembered, for each state j and each time t, the source of maximum value of $\Phi_i(t-1)a_{ij}$ (i.e. "from where we came"), we can back-trace the optimal state sequence $X^\star$ from the end (similarly as for DTW). Figure 6 shows the possible paths through the model of our example. The state sequence $X = [1, 2, 2, 3, 4, 4, 5, 6]$ used as example in subsection 2.4 is shown in red.

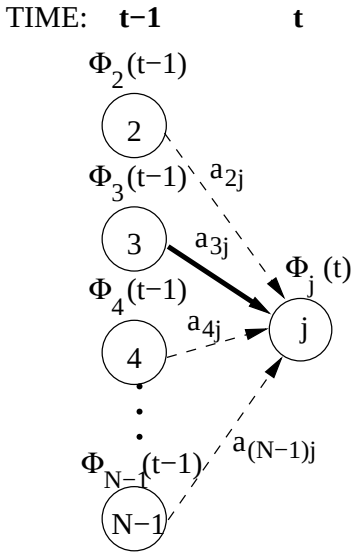

Figure 11: Computation of partial VITERBI probability $\Phi_j(t)$. The bold line denotes the selected maximum, the dashed lines the discarded paths from other states.

6 Continuous speech recognition

The models are "glued" each to other using the non-emitting states (their purpose is to provide this seam-less "glue"). One "mega-model" is created in this way. The language knowledge (in the form of transition probabilities between phonemes or words) can be built in this model.

For an input sequence, we find the optimal way through this mega-model. When we finish, we can back-trace the path and the units found on this path (phonemes, words) are the output of the recognizer.

References

[1] L. Rabiner and B.H. Juang. *Fundamentals of speech recognition*. Signal Processing. Prentice Hall, Engelwood Cliffs, NJ, 1993.

[2] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE*, 77(2):257–286, February 1989.

[3] S. Young, J. Jansen, J. Odell, D. Ollason, and P. Woodland. *The HTK book*. Entropics Cambridge Research Lab., Cambridge, UK, 1996.

[4] S.J. Young, N.H. Russell, and J.H.S. Thornton. Token passing: a simple conceptual model for connected speech recognition systems. Technical Report CUED/F-INFENG/TR.38, Cambridge University Engineering Department, July 1989.