REDUCED COMPLEXITY AND SCALING FOR ASYNCHRONOUS HMMS IN A BIMODAL INPUT FUSION APPLICATION

Marc Al-Hames and Gerhard Rigoll

Technische Universität München
Institute for Human-Machine Communication
Arcisstrasse 16, 80333 München, Germany
{alh, rigoll}@mmk.ei.tum.de

ABSTRACT

The Asynchronous Hidden Markov Model (AHMM) can model the joint likelihood of two observation sequences, even if the streams are not synchronised. Previously this model has been applied to audio-visual recognition tasks. The main drawback of the concept is its rather high training and decoding complexity. In this work we show how the complexity can be reduced significantly with advanced running indices for the calculations. Yet, the AHMM characteristics and its advantages are preserved. The improvement also allows a scaling procedure to keep numerical values in a reasonable range. In an experimental section we compare the complexity of the original and the improved concept and validate the theoretical results. Then the model is tested on a bimodal speech and gesture user input fusion task: Compared to a late fusion HMM an improvement of more than 10% absolute recognition performance has been achieved.

1. INTRODUCTION

The Hidden Markov Model (HMM) is a popular statistical tool, that has been applied successfully to a wide range of problems, especially automatic speech recognition [1]. The Asynchronous Hidden Markov Model (AHMM) [2] is an extension of this concept that can model the joint likelihood of two potentially asynchronous observation streams. It is related to Input-Output HMMs [3], but only very loosely connected to the homonymous concept proposed in [4].

The AHMM addressed in this work is used to learn and model the likelihood of two observation streams describing the same event class. The model has been applied successfully to various audio-visual recognition tasks, like speech recognition [2], person identification [5], or meeting analysis [6]. It can generalise the connection between the streams, even if they are not synchronised. This is the advantage of the AHMM compared to other multi-modal Markov models, like early fusion or coupled HMMs. The main drawback of the concept is its high complexity for both training and decoding. For long sequences the model can become computational infeasible.

In this work we apply advanced running indices during the calculations. This way the complexity can be reduced to a high degree. On the other hand the developed method is still mathematical exact, as only unnecessary points will be omitted. With this improved procedure the AHMM concept can be applied to a wider range of problems, where e.g. the streams are very long or have a very high degree of asynchronity. Furthermore the new indices also allow the development of a straightforward scaling procedure similar to scaling in standard HMMs [1]. Thus numerical values can be kept in a reasonable range during calculations.

One problem with a very high time asynchronity between different channels are multimodal systems [7]. They are designed, such that the human-machine interaction is more natural and more intuitive for the user. Several input devices can be used either synchronous or sequentially. Furthermore users may switch between the different devices to achieve the same system reaction. The challenge of these systems is to fuse the different inputs - both in time and between the different channels - to meaningful commands. Especially the bimodal combination of speech and gesture has been deeply investigated in the last 20 years. One famous example is Bolt's "Put-that-there" system [8]. Several approaches for the fusion process have been developed since then, among them rule-based, statistical, and hybrid methods [7, 8, 9, 10].

Most of these methods require a large domain knowledge to actually model the problem. In the experimental section of this work we will show how the AHMM can be applied to the problem of bimodal speech and gesture input fusion, without using domain knowledge. This way the system can easily be extended to different input devices or applications.

2. THE ASYNCHRONOUS HMM

The AHMM is used to model the joint likelihood $p(\vec{x}, \vec{y})$ of two observation sequences $\vec{x}$ with length T, and $\vec{y}$ with length S. Without loss of generality it is assumed that $S \leq T$ (if $\vec{y}$ is longer than $\vec{x}$ an extension of the concept is necessary). The joint likelihood can of course not be calculated directly, as it is intractable.

Therefore two hidden variables are introduced: the first variable $q_t = 1 \dots N$ is synchronised with the stream $\vec{x}$ and identical to the state in standard HMMs. The total number of states in the model is denoted as N. In the AHMM concept it is assumed, that a state always emits a symbol from the stream $\vec{x}$ at each time step t. Furthermore each state $q_t = i$ can with the probability $\epsilon(i, t)$ at the same time emit a second symbol from the stream $\vec{y}$. The hidden variable $\tau_t = 0 \dots S$ models the alignment between $\vec{x}$ and $\vec{y}$. Whenever a state emits a symbol from stream $\vec{y}$, the alignment variable τ_t is incremented, until all symbols from $\vec{y}$ have been emitted and $\tau_t = S$.

This can be represented in a three dimensional trellis, as shown in Fig. 1 (left). The two axis time t and state q are identical to those from HMMs. The axes s represents the alignment between the streams. In each time step t one symbol from $\vec{x}$ is emitted. This corresponds to move one step right in the trellis. If a state also emits a symbol from $\vec{y}$, the movement is one step right and one step up. As a state always emits a symbol from $\vec{x}$ a movement up without a movement right is not possible. The model can of course in each step jump to any of the states from the state axes (for an ergodic model).

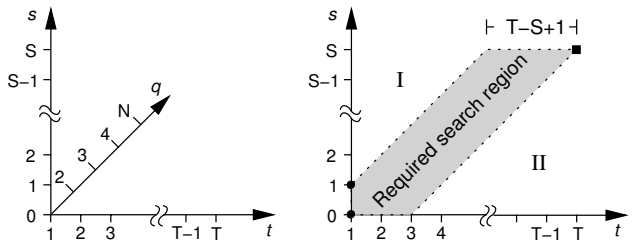

Fig. 1. 3D trellis (left) with time *t*, alignment *s*, and state *q*. 2D projection of this trellis (right), where the state axis is omitted. The 2D trellis shows the region that has to be calculated. Only points in this area can actually lead to a valid path through the model.

The AHMM can then be parameterised with five distributions:

- The initial state distribution: $\pi_i = P(q_1 = i)$
- The state transition distribution: $A_{ji} = P(q_{t+1} = i | q_t = j)$
- The probability of emitting two symbols in a state. For simplification, the time index will be omitted in this distribution: $\epsilon_i = P(\tau_t = s | \tau_{t-1} = s-1, q_t = i)$
- The emission distributions for a single symbol $P(\vec{x}_t | q_t = i)$ and for a pair of symbols $P(\vec{x}_t, \vec{y}_t | q_t = i)$.

As with standard HMMs, the emission distributions can be modelled discretely or continuously (e.g. with a mixture of Gaussians).

2.1. Likelihood computation

To compute the joint likelihood $p(\vec{x}, \vec{y}|\lambda)$ of two streams, a forward procedure has been developed in [2]. At first this forward procedure shall be recapitulated in detail without the borders for the running indices. The forward path variable is *defined* as:

$$\alpha(i, s, t) = p(q_t = i, \tau_t = s, \vec{x}_t, \vec{y}_s)$$

The model can start with either emitting one or two symbols in the first step. The *initialisation* step therefore is:

$$\alpha(i, 0, 1) = [1 - \epsilon_i] \cdot \pi_i \cdot p(\vec{x}_1 | q_t = i) \quad (1)$$

$$\alpha(i, 1, 1) = \epsilon_i \cdot \pi_i \cdot p(\vec{x}_1, \vec{y}_1 | q_t = i) \quad (2)$$

As long as none of the symbols from $\vec{y}$ have been emitted ($s = 0$), the *induction* step is:

$$\alpha(i, 0, t+1) = [1 - \epsilon_i] \cdot p(\vec{x}_{t+1} | q_{t+1} = i) \sum_{j=1}^{N} p(q_{t+1} = i | q_t = j) \cdot \alpha(j, 0, t) \quad (3)$$

If a symbol from $\vec{y}$ has already been emitted ($s > 0$), the *induction* step becomes:

$$\alpha(i, s+1, t+1) = \epsilon_i \cdot p(\vec{x}_{t+1}, \vec{y}_{s+1} | q_{t+1} = i) \sum_{j=1}^{N} p(q_{t+1} = i | q_t = j) \cdot \alpha(j, s, t) + [1 - \epsilon_i] \cdot p(\vec{x}_{t+1} | q_{t+1} = i) \sum_{j=1}^{N} p(q_{t+1} = i | q_t = j) \cdot \alpha(j, s+1, t) \quad (4)$$

Finally the *termination* with the likelihood of the observations is:

$$p(\vec{x}, \vec{y}|\lambda) = \sum_{j=1}^{N} \alpha(j, S, T) \quad (5)$$

The original AHMM algorithm does not explicitely state the borders for the running indices i, s, and t in the forward path calculation. But the model complexity is given as $\mathcal{O}(N^2 ST)$. Each induction step approximately requires N summations. Therefore the original complexity is derived, if the forward variable α in Eq. (3) and (4) is calculated for each combination of $1 \leq i \leq N$, $1 \leq t \leq T$, and $0 \leq s \leq S$. Fig. 1 (right) shows the AHMM trellis, where the state axis is omitted. In this representation the algorithm corresponds to calculating each point within the (s, t)-plane (and of course one plane for each state i). However this is not necessary.

In Fig. 1 (right) the initialisation step is represented with the two black circles: the model can only start with emitting one, or two symbols. The termination step Eq. (5) requires, that all symbols from $\vec{x}$ ($t = T$) and all symbols from $\vec{y}$ ($s = S$) have been emitted. In Fig. 1 this point is represented with the black square. Consider first the region marked I in Fig 1. At each time step t a symbol from $\vec{x}$ is emitted, and with the probability ϵ_i a second symbol from $\vec{y}$. Thus the model can at no time had emitted more symbols from $\vec{y}$ than from $\vec{x}$, or mathematically $s \leq t$; but in the first region $s > t$. With the original forward procedure the values in this region are calculated and will in general be $\alpha > 0$, but they will never be reused for the termination step. The calculation of points in region I is therefore not required. Now consider region II. Again recapitulate that the model in each time step emits a symbol from $\vec{x}$, it will never emit only a symbol from $\vec{y}$. It is therefore not possible to move only up, but only right, or right-up in the trellis. From the points in region II the termination point can therefore not be reached anymore. Again the original algorithm calculates the points in this region, but they will not be reused for the termination step. Thus the calculation of points in region II is not required. In summary only points in the grey region in Fig. 1 can actually end up in the termination point. Thus it is only necessary to calculate the α-values in this region.

This can be used to derive advanced limits for the running indices in the forward procedure. The α-values in Eq. (3) then only have to be calculated for:

$$1 \leq i \leq N \text{ and } 1 \leq t \leq T - S \quad (6)$$

and the α-values in Eq. (4) only have to be calculated for:

$$1 \leq i \leq N, 1 \leq t \leq T, \text{ and } \max(0, t - (T - S)) \leq s \leq \min(S, t) \quad (7)$$

The same technique can be applied to Viterbi-decoding and to the backward calculation (Sec. 2.4).

2.2. Complexity

The original AHMM algorithm [2] complexity is $\mathcal{O}(N^2 ST)$, where again N is the number of total states in a fully connected ergodic model and T and S the lengths of the two observation streams. This can become quickly computational infeasible if the streams are too long. In [2] it is suggested to force the alignment between $\vec{x}$ and $\vec{y}$, such that $|t - \frac{T}{S}| < k$, with k some constant indicating the maximum allowed stretching between the two streams. Then the complexity of the model is reduced to $\mathcal{O}(N^2 Tk)$. This however narrows the search space to a high degree and therefore limits the advantages

of the AHMM concept. This is applicable for audio-visual recognition problems, where the asynchronity between the audio and visual stream can indeed be assumed to be limited to a couple of frames. Yet, for other problems this is not practical, as the two streams may have a strong asynchronity that can furthermore vary from sample to sample. Thus a fixed stretching factor can not be determined.

With the proposed improvement in this work only points in the forward- and backward-path, that will actually be reused in later steps, are calculated. Thus unnecessary calculations are avoided. Yet the result will still be exact, as the omitted points will never lead to a valid path through the trellis. Using this advanced procedure, the calculation time of the AHMM can be reduced to

$$\mathcal{O}(N^2[TS - S^2 + T])$$

Depending on the length of the two streams, the reduction of necessary calculations compared to the original algorithm is between 0 (if $S = 1$) and N^2T^2 (if $T = S$). In the worst case one sequence has double the length of the other ($T = 2S$). Then the maximum number of calculations for the optimised procedure is $N^2(\frac{T^2}{4} + T)$. Yet in this case the optimised procedure still requires $N^2\frac{T^2}{4}$ less calculations than the original procedure. At the same time the exact likelihood result is achieved, and no restriction of any kind on the asynchronity between the streams is set. This advanced procedure can therefore be applied to a wider range of problems.

2.3. Scaling

Despite the reduction in complexity, the new running indices for t and s in the forward path calculation Eq. (1) - (5), also allow a simple and exact scaling procedure. This enables a computation of the model without numerical problems [1]. An exact scaling is not possible for the original calculation procedure. To derive scaling values for each time step t all $\alpha(i, s, t)$ values in the (q, s)-plane have to be summed up after each time step. In the original algorithm some of these values are $\alpha > 0$, but they will never lead to a valid path through the model. Thus, if they are still be summed up for the scaling procedure, the final complete likelihood is effectively multiplied with probabilities that have no influence on the observation likelihood $P(\vec{x}, \vec{y}|\lambda)$. Therefore the final derived likelihood value $P^o(\vec{x}, \vec{y}|\lambda)$ will be to low compared to the exact $P(\vec{x}, \vec{y}|\lambda)$. This can only be avoided, if just the valid points are summed up. Yet, this leads to the procedure proposed in this work. However, the error made by the wrong scaling has no influence on recognition results, as it is systematic among all models. But it is not mathematical correct, and the correct observation likelihood can never be calculated.

This can be avoided with the running indices introduced in this work. Here only α-values that will indeed result in a valid path through the model are calculated. For each time step t all valid α-values, according to Eq. (7) are calculated. If the highest calculated alignment s in the time step t is referred to as s_t, the scaling coefficient can be calculated as:

$$s(t) = \frac{1}{\sum_{s=0}^{s_t} \sum_{i=1}^{N} \alpha(i, s, t)} \tag{8}$$

Then the scaling procedure is like in usual HMMs [1]: The scaling has to applied after each time step and all α values at time t are scaled according to $\tilde{\alpha}(i, s, t) = s(t) \cdot \alpha(i, s, t)$. In the next time step the α-values are now calculated with $\tilde{\alpha}$ instead of α. Finally the exact log-likelihood of an observation can be calculated as

$$\log P(\vec{x}, \vec{y}|\lambda) = \sum_{t=1}^{T} \log s(t) \tag{9}$$

With the proposed indices in this work we can therefore apply an exact scaling procedure to the AHMM and keep the numerical values in a computational reasonable range.

2.4. Viterbi Decoding and EM Training

A Viterbi decoding and an Expectation-Maximisation (EM) training algorithm for the AHMM have been derived in [2]. The Viterbi decoding is similar to the forward path calculation, but the summations are replaced by maximisations. Then the best state-sequence and the best alignment of the two streams can be obtained with backtracking.

For the EM-algorithm, the maximisation step is identical to the one in standard HMMs. For the expectation step a backward variable β and two parts of the forward variable are required: namely, where a state emits only one symbol α_0, resp. two symbols α_1. They are derived in the same fashion as the forward path and described in [2].

Yet, again the running indices can be improved with Eq. (6) and (7), both for the Viterbi procedure and for the expectation step during EM. Thus the improvements proposed in this work reduce both the complexity of training and decoding.

3. EXPERIMENTS

To investigate the actual difference in complexity between the original and the optimised method proposed in this work, both versions have been implemented. The required CPU ticks for decoding have been measured for different combinations of stream lengths. Fig. 2 shows an exemplary result of the required decoding time for both approaches. In this example the AHMM has five ergodic hidden states and discrete output probabilities. The length of the $\vec{x}$ stream has been fixed to $T = 250$, the length of the $\vec{y}$ stream has been varied between 1 and 250. In general, the measurements validate the theoretical result, but there is one minor difference: In theory the complexity of the optimised strategy should always be less, or at least equal to the original strategy. However for very short $\vec{y}$ sequences, where $S/T < 0.05$, the advanced strategy can lead to a slightly higher decoding complexity. The advanced strategy requires to calculate the running indices for each induction step. This produces an overhead with a fixed number of calculations, with less than a couple of thousand CPU ticks. It has only an influence, when the decoding time is very small anyway. Thus this overhead has no practical impact.

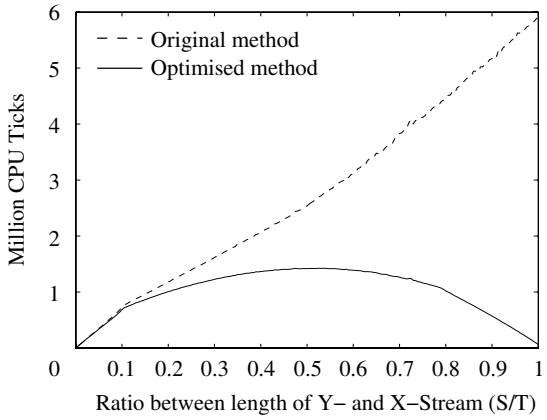

Fig. 2. Measured decoding complexity.

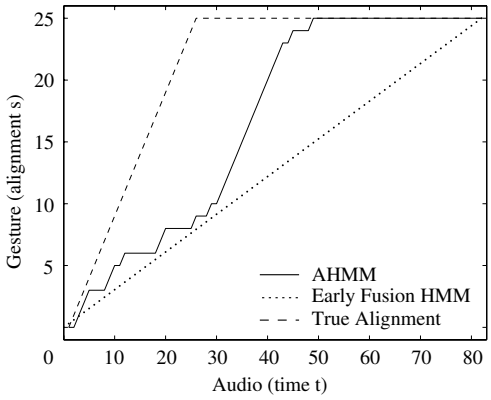

Fig. 3. Stream alignment for an HMM and AHMM

In a second experiment the optimised AHMM has been applied to the problem of fusing bimodal speech and mouse-gesture input. Our system uses 11 words in the speech stream and 10 mouse gestures. Both streams combined form 25 higher level system commands. The information from both channels is required to form a valid command. Each command can furthermore be generated with different combinations of speech and/or gesture. Our data set comprehends samples from 1872 bimodal system commands. Each command can either be synchronous, partly overlapping or sequential. The data set is split into 1240 samples for training and verification and 632 samples for test. The test samples are partly from unknown and partly from known persons. From the speech stream 12 MFCCs plus the energy and the Δ and $\Delta\Delta$ were extracted. This set was then discretised with a codebook size of 100 symbols. The recognition rate for the isolated words in the audio stream with a discrete HMM is 92.48%. For the gesture stream, one of eight possible directions is stored whenever the mouse is moved. The recognition rate for the individual gestures is 89.90% with a discrete HMM.

Then the AHMM has been applied to recognise system commands: One AHMM for each command has been trained with the bimodal data. For comparison an early and a late fusion HMM have been trained as well. For the early fusion the sample-rates of the two streams have been adjusted and both streams concatenated to one large feature stream. For the late fusion, both streams were classified with individual HMMs and the recognition results later fused with a multiplication of the N-best-lists. Fig. 3 shows an exemplary alignment between the two streams, as well as the alignment obtained by early fusion and the AHMM. Both gesture and speech start at $t = 0$, then the gesture ends at $t = 24$ and the word is continued. The AHMM does not match this alignment perfectly, as it in this example emits the gesture stream slower; but the general characteristic of the bimodal input is preserved. The early fusion HMM can not model this alignment. It simply emits two symbols in each step.

This has an impact on the performance of the early fusion HMM: several configurations have been tested, but the concept was not able to generalise the high asynchronity in the data. The achieved system command recognition rates are only slightly better than guessing. With the late fusion the data can be modelled much better. Several combinations of individual HMMs with different states have been tested. The recognition rates reach up to 67.19% percent. This is outperformed by the AHMM: The capability to model the asynchronity leads to a system command recognition rate of 77.62%. Thereby AHMM decoding can be performed in real-time. The main results for the late fusion and the AHMM are summarises in Tab. 3.

AHMM		**Late Fusion HMM**		
States	Rate	Word Stat.	Gesture Stat.	Rate
5	73.66%	10	5	64.20%
10	77.08%	10	10	60.86%
25	77.62%	15	5	67.19%

Table 1. Recognition results for the AHMM and a late fusion HMM. For the late fusion different combinations of states for both the word and the gesture HMM have been tested.

4. CONCLUSION

In this work we improved the AHMM concept by introducing advanced running indices for the forward and backward calculations. Only necessary points in the trellis are actually calculated. This way the complexity of both coding and decoding has been reduced significantly. Yet, the achieved result is still exact. With the proposed improvement we then derived a scaling procedure for the AHMM that keeps numerical values in a computational reasonable range.

In an experimental section we validated the theoretical reduction in complexity. Then the AHMM has been applied to the task of combined speech and gesture input. The AHMM outperforms a late fusion HMM by more than 10%. However the absolute recognition performance is not sufficient for an application. Further improvements - especially on the features - are required.

Yet the results in this work show, that the AHMM can not only be applied to audio-visual recognition problems, but also to tasks where the streams can show a very high degree of asynchronity.

5. REFERENCES

[1] L.R. Rabiner, "A tutorial on Hidden Markov Models and selected applications in speech recognition," *Proc. of the IEEE*, vol. 77, no. 2, pp. 257–285, February 1989.

[2] S. Bengio, "An asynchronous Hidden Markov Model for audio-visual speech recognition," in *Advances in NIPS 15*, Cambridge, MA, 2003.

[3] Y. Bengio and P. Frasconi, "An input-output HMM architecture," in *Advances in NIPS 7*, Cambridge, MA, 1995.

[4] A. Garg, S. Balakrishnan, and S. Vaithyanathan, "Asynchronous HMM with applications to speech recognition," in *Proc. IEEE ICASSP*, Montreal, Canada, 2004.

[5] S. Bengio, "Multimodal authentication using asynchronous HMMs," in *Proc. IEEE AVBPA*, 2003.

[6] D. Zhang et al., "Modeling individual and group actions in meetings: a two-layer HMM framework," in *Proc. IEEE CVPR*, 2004.

[7] S. Oviatt, "Multimodal interfaces," in *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*, 2003, pp. 286–304.

[8] R.A. Bolt, "Put-that-there: Voice and gesture at the graphics interface," in *Proc. SIGGRAPH*, 1980.

[9] F. Althoff, M. Al-Hames, G. McGlaun, and M. Lang, "Towards a new approach for integrating multimodal user input based on evolutionary computation," in *Proc. IEEE ICASSP*, Orlando, USA, 2002.

[10] S. Oviatt et al., "Toward a theory of organized multimodal integration patterns during human-computer interaction," in *Proc ICMI*, 2003.