Application of graphical models
Audio-Visual Speech Recognition

Mathilde BATESON
mathildebateson@gmail.com

Vivien CABANNES
vivien.cabannes@ens.fr

***Abstract*—Audio-Visual speech recognition is an instance of a signal collaboration problem. It involves audio recognition, computer vision and time series modelling. The main framework is rather intuitive: getting task-related audio and visual descriptors, before modelling the relationship between those descriptors and the word that has been pronounced.**

The first step consists in extracting descriptors. In order to exploit the temporal structure, descriptors should keep a time dimension. Natural descriptors are mouth movements through lips position, and audio local information through scalograms.

The second stage consists in time-series modelling. It will be tackled through graphical modelling, with refined hidden Markov models, hidden nodes designed to point towards a phoneme position (*ex.* fourth phone in the word), and capture its class (*ex.* explosive, fricative). Several architectures are proposed to enhance stream collaboration.

***Keywords*—speech recognition, audio processing, image segmentation, time-series, hidden Markov model, signals collaboration**

I. INTRODUCTION

A. Motivations

Amongst the widely researched artificial intelligence problems is designing systems able to recognize speech. Although human understanding is mainly based on listening, it can be enhanced by looking at a face expression or lips reading. More generally, addressing the problem of correlating information from different signals has many practical applications.

B. Problem

This project tackles the task of recognizing a word pronounced in a audio-video sample. The problem is split into a recognition stage based on a mono-stream signal (only audio, or only video), and a stage of correlating between signal streams. The mono-stream signal recognition system is itself split in getting good descriptors, and in modelling the relationship between those descriptors and the word that has been said.

C. Dataset

To tackle this problem, let's look at few words pronounced in uncluttered condition. A useful dataset has been collected by Fu Jie Huang, described *here*. It is made of 78 words pronounced ten times by three women and six men. Audio data is given under WAV file. Video data is given through six numbers describing the mouth at each frame. Segmentation between words are indicated in a specific files.

D. Related work

Huang models an observation at time t by Gaussian mixture model (GMM), before adding an hidden Markov model (HMM) behind timed observation. Their work is resumed in [3]. In fact, the use of HMM had already been investigated for such a task, as illustrated by Rabiner in its introduction on HMM for speech recognition [6].

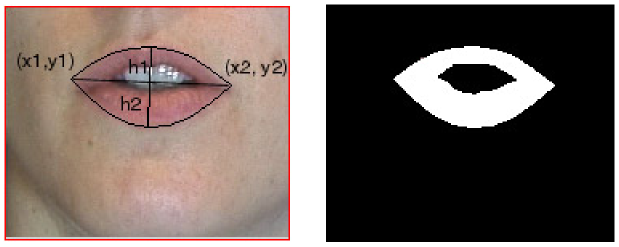
In [5], Murphy et al. followed this idea, looking more precisely into capturing streams dependency. They have used ideas described by Ghahramani in [2] and Brand in [1] on so-called factorial and coupled HMM (F-HMM and C-HMM). It is in this last paper that are explained the approaches we are investigating.

II. DESCRIPTORS

Tackling the problem will be facilitated by considering smart descriptors as input. Those descriptors should be low-dimensional and contain a maximum of discriminative information. We are seeking to understand sources of variability before removing the non-discriminative ones (*e.g.* pitch of voice). Mathematically, it can be seen as invariant quotient.

A. Video motion

Descriptors should mainly capture lips motion. Six simple descriptors are proposed by the dataset authors, as shown in figure 1(a), corresponding to labial corners position and two indexes for mouth opening. However, one could think of more adequate descriptors, such as a black-and-white map corresponding to lips segmentation; or even better, a grayscale relief map to better capture lips motion.



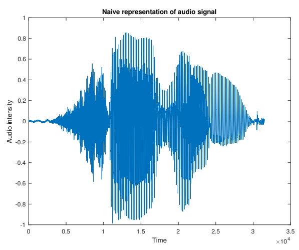
((a)) dataset descriptor ((b)) smarter descriptor

Fig. 1. Extracting video descriptors.

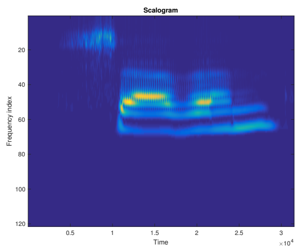
Rotation and translation invariance suggests to only keep the distance between the two lip corners, rather than the four coordinates (x_1, x_2, y_1, y_2), making mouth descriptors only three-dimensional.

B. Audio signal

Seeking a sparse and sequential audio descriptor naturally calls for a window-Fourier transform, or it refinement in wavelets[1].We used wavelet representations (often called a scalogram).



((a)) Naive representation



((b)) Sparse scalogram

Fig. 2. Different representation of the audio signal "seven".

Yet the scalogram is too high dimensional regarding implementation constraints. As one can see, only a dozen of coefficients could summarize its structure at a given time: positions and values of significant local maxima.

Once this new descriptions has been made, one can still try to reduce dimensionality by capturing invariance: *e.g.* since this problem is not regarding the pitch of the voice, one can quotient by the group of transposition.

III. Modelling

Given some descriptors, we want to build a model which outputs a prediction for the word which has been said.

A. General architecture

Let's take a generative point of view. To an observation is added a latent variable corresponding to the word which has been said. The main problem is to model an observation knowing which word has been said.

It is natural to cut a word into basic unit, such as phones or syllables, to model each of those unit, before reconstructing a word from those units. This scheme is particularly useful to reduce dimensionality of the learning.

However, for a small dictionary, it is reasonable and practical to skip this step and consider words as basic units. Thus, this paper won't focus on this part, yet our architecture will be designed in a way that tries to catch this hierarchy.

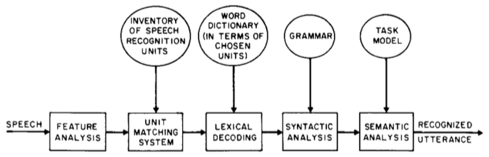


Fig. 3. Speech recognizer architecture. © [6]

[1]Some standard descriptors have arisen in literature [7] such as MFCC descriptors. However, we thought best to think from scratch.

B. Proposed model

Dynamic graphical models based on hidden Markov model have proven well suited to capture sequential dependency. They realize a good trade-off between model complexity and ability to do computation.

In our modelling[2], we add to the sequential signal a hidden left-to-right Markov chain to capture the division into more basic units (*i.e.* an increasing latent variable corresponding to which unit we are in at a given t). Here, five states will be considered, yet an improvement would be to adapte this for different words.

Given a word and the latent variable saying which state we are in, we expect to observe the realization of a basic unit (phoneme, syllabus or so on), thus we can expect mainly three states: silence, attack, ending. It will be captured into an Gaussian mixture model with three components.

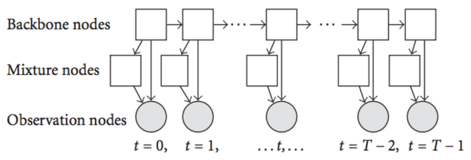


Fig. 4. The proposed model is made of a left-to-right hidden Markov chain. A mixture node allow to build on top of it a mixture model. © [5]

C. Stream collaboration

Up to now we have focus on recognition from a mono-source signal. Let's now attack the audio-video signal collaboration. We will refer to each signal as a stream.

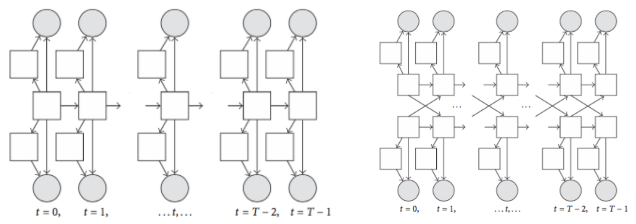
A simple solution consists in concatenating descriptors. However, there is plenty other solutions to imagine stream collaboration. The idea is to design one HMM per stream but add some variables sharing or interactions.

It seems that the state acts like a pointer towards important stages of the word. In consequence, streams state should be equal or at most shift of one (mouth opening before a sound is produced). Those ideas are respectively captured by the multi-stream HMM (MSHMM [4]) and the coupled HMM (CHMM [1]) which have shown best results [5]. The MSHMM assumes that the audio and video sequences are state synchronous, which is a simplification, but allows the observation likelihood in each stream to be computed independently. A natural extension is the CHMM, with the same independence between observation likelihood, but where the backbones nodes interact, bringing in some asynchrony between the streams.

IV. Learning

Learning in generative models is usually performed via likelihood maximization. Because of the model complexity, relaxations will be useful, involving EM algorithms for latent variables (backbone and mixture nodes).

[2]Equations lovers might refer to section Learning for further calculation.



((a)) Two-stream HMM ((b)) Coupled HMM

Fig. 5. Different collaborations between hidden Markov models.

A. *Parametrization*

Let's introduce o the observation, w the corresponding word, q the backbone node, and c the mixture component. The recognition algorithm outputs

$$w_{\text{pred}}(\bar{o}) = \arg\max_{w} p_\theta(w|\bar{o}),$$

Yet θ has to be determined, it is fully parametrized by

$$\begin{aligned}
&p(w) = f_w,\\
&p(q_t = j|q_{t-1} = i, w) = a_w(i,j),\\
&p(c_t = m|q_t = i, w) = s_{w,i}(m),\\
&p(o_t|c_t = m, q_t = i, w) = \mathcal{N}(o_t, \mu_{w,i,m}, U_{w,i,m}),
\end{aligned}$$

Fitting those parameters is an instance of supervised learning, it is made via maximization of $p(o, w; \theta)$ with respect of θ. It can be made word independently since

$$p(o, w; \theta) = p(w; f)p(o|w; a, s, \mu, U),$$

The first term is just a frequency, the second corresponds to the hidden Markov model. Let's focus on its maximization.

B. *Fitting a hidden Markov model*

The distribution $o|w$ is well explained by the latent variables q and c. This invites to use a relaxation by guessing the variable q and c, before maximizing the joint likelihood, and to iterate until a convergence criterion is reached.

If the prediction at the end should be the one which reduces the most the relaxation gap, the loops can be speed up at the beginning at the beginning with bigger steps. A simple way to do so is the following algorithm:

1) Initialize q uniformly, and c with the k-means
2) Perform *EM* but with a deterministic *E-step*: $p(q, c) = \delta_{q_0,c_0}$, where q_0 and c_0 are given by the mode.
3) Perform general *EM*

Let's now derive those *EM* algorithms for our models. The *E-step* is driven by quantities we need to derive for the *M-step*, so let's focus on it first.

C. *M-step*

Following calculation in appendices, the *M-step* reads

$$\begin{aligned}
&a_{i,j} = \frac{\sum_{r,t} \eta_{r,t}(i,j)}{\sum_j \sum_{r,t} \eta_{r,t}(i,j)}, \quad s_{i,m} = \frac{\sum_{r,t} \gamma_{r,t}(i,m)}{\sum_m \sum_{r,t} \gamma_{r,t}(i,m)}\\
&\mu_{i,m} = \frac{\sum_{r,t} \gamma_{r,t}(i,m) o_{r,t}}{\sum_{r,t} \gamma_{r,t}(i,m)}\\
&U_{i,m} = \frac{\sum_{r,t} \gamma_{r,t}(i,m)(o_{r,t} - \hat{\mu}_{i,m})(o_{r,t} - \hat{\mu}_{i,m})^T}{\sum_{r,t} \gamma_{r,t}(i,m)}
\end{aligned}$$

where r index the samples.

D. *E-step*

The *M-step* has shown two quantities to derive during the *E-step*, η and γ. Those quantities are derived following usual derivations on HMM and GMM. However, we didn't address the problem of learning a HMM model with several examples, but chose the simplest way to do it, through basic averaging. However, learning with several examples seems an interesting topic, with some issues that could lead to implement ideas such as mini-batch.

E. *Prediction*

Given a sample o, the algorithm naturally outputs $w = \arg\max_w p(w|o)$, which is $\arg\max_w p(w)p(o|w)$. The derivation of $p(o|w)$ implies to sum on all latent variables, which is not really suitable. A simpler solution is to approximate once again with the classical *EM* relaxation

$$p(o|w) \approx \mathbb{E}_{q,c|o}[p(o, q, c|w)]$$

These approximations made the computations easier for our word retrieval algorithm.

V. Implementation & Results

The first task consists in preprocessing the audio file and link them with the video text descriptors, before getting a proper input.

A. *Descriptors*

Given the dataset, there is not so much work to be done on video descriptors, mainly read text files, and get the three dimensional feature described before.

The first main task consists in preprocessing audio files to get useful descriptors. Our implementation reads

The matlab built-in implementation is really slow, involving a non-optimized loop in t and functions `cwt`, `findpeaks`. It takes days to get all the dataset preprocessed.

Algorithm 1: Extracting audio descriptors

```
Data: Audio file, $n$ given by the user
1 Perform wavelet transform $F(t, \omega)$;
2 Threshold to remove noise;
3 for $t = 1 \dots T$ do
4     Extract $(\omega_{t,i})_{i \leq n}$ the largest peaks of $F(t, \omega)$;
5 end
6 return $(\omega_{t,i}, F(t, \omega_{t,i})_{i \leq n, t \leq T}$;
```

B. Difficulties

Several difficulties had to be tackle: write a simple HMM, write a refined HMM, with the mixture node, learn the HMM with several observations, build stream-collaborative HMM; before learning these models on few words, and trying to predict new words. In other terms, our nice simple theoretical ideas have to wait in line to be implemented, unfortunately programs are slow and adding debugging doesn't help.

Implementation had to deal with exponential vanishing through taking logarithm.. An other problem is, depending on the descriptors, the fitted Gaussians do not necessarily have densities, which means that the empirical covariances are not invertible. One should consider projecting points on the range of the covariance matrix, and penalizing the distance between the point and this space, which amounts to hand-building a new distribution as a product of a Gaussian and a distribution defined by the penalization.

C. Our results

The program is really slow, actually classic implementation uses approximate sum-product as described in [2]. However, our video descriptors are small enough to run the entire program, with the results shown below:

TABLE I. EXPERIMENTAL RESULTS

Predictions precision	video	audio
train	10%	too slow
test	30%	too slow
random	1.5%	too slow

Some instabilities of the programs have not been solved yet, probably due to silence sequence, where the mixture is crushed as a Dirac, and all the Gaussians return an error treated as "NaN", before propagating this error in the state, and finally in the whole likelihood being computed as "Not a Number".

D. Literature results

This project was based on the article [5], which doesn't present in detail its algorithm, focusing on stream collaboration in a HMM, and audio noise. The authors don't present in detail how they computed their precision rate, however, they yield really good results with

TABLE II. LITERATURE RESULTS

	video	audio	Multi-stream HMM
Precision	67%	97%	99%

They also show numerically that in presence of audio-noise, the stream collaboration allows to consequently increase precision.

VI. For a future work

There is a lot to be done to complete this attempt at using HMM for audio-visual speech recognition. A first idea would be to find ways to accelerate the HMM learning through approximation in the E-step. This would probably be part of a more general study on EM acceleration through approximation in graphical models. Then would be tackled the collaboration stream. After which could be consider cluttered conditions, with artificial noise, or more complex problems, like real sentences from movie.

Yet, behind this HMM framework, a lot of other ideas emerged from this project.

A. Descriptors

In this work, we wanted to keep the sequential data, so each time frame were keppt, which has implied way to big descriptors: 28000 frames per word on average. However, their is a lot of redundancy in those descriptors, so to reduce dimensionality and speed up the learning, a local averaging would be of great use.

Also, going back to the speech recognizer architecture figure 3, our descriptors allow use to easily find silent parts in the word, and so to extract more basic units, which would correspond to phonemes for example. We could them model those more basic units with a HMM, or a simpler model.

B. Model

In fact, HMM was introduced because it is a complex enough model with capacity to do fast computation, as neural networks now being introduced everywhere. Yet, there is a lot of reason to think for something else. First, it seems reasonable, since it is hard to model observations, to choose a discriminative framework. Neural networks can be seen as the discriminative version of a complex enough model with capacity to do fast computations. And we could guess that the current state-of-the-art audio-speech recognition systems are designed with such architectures.

An interesting thing to do, would be to look for the discriminative model generated by this HMM framework, as linear discriminative analysis generates logistic regression, or quadratic discriminative analysis generates logistic regression with a quadratic kernel.

C. Concrete improvement

A simple idea would be to cut our audio descriptors according to silent parts and explicit the latent state variable corresponding to this cut. We could then model each observation given a state by a simple hidden Markov model, with like five states corresponding to five Gaussian distribution.

A nice progress would be to introduce some discriminative model at some point. For example, once our generative HMM has been learnt, rather then comparing probabilities $p(o|w)$, one could compute the mode estimation of latent variables, before giving it as an input to a neural network to perform final classification.

VII. Conclusion

This paper attempted to perform audio-visual speech recognition using tools from probabilistic graphical modeling. This framework allows efficient stream collaboration, through hidden variables interactions. A good understanding of the problem helps to design smart low-dimensional descriptors and reduce the complexity of the forward modelling. Here, too big audio descriptors were used, resulting on a really slow algorithm, and a need for faster the sum-product algorithm through, for example, structural approximation.

Appendix A
HMM Equations

Let's begin with a mono-stream HMM, the multi-stream will follow the same derivation with one more index for the stream.

A. *M-step*

The joint log-likelihood reads for a sample

$$\log\, p(o,q,c) = \sum_{t=1}^{T-1} \log p(q_{t+1}|q_t) + \sum_{t=1}^{T} \log p(c_t|q_t) + \sum_{t=1}^{T} \log p(o_t|c_t,q_t)$$

Introducing binary vectors and precedent parametrization

$$\log\, p(u,q,c) = \sum_{t,i,j} q_t^i q_{t+1}^j \log(a_{i,j}) + \sum_{t,i,m} q_t^i c_t^m \log(s_{i,m}) + \sum_{t,i,m} q_t^i c_t^m \log(\mathcal{N}(o_t,\mu_{i,m},U_{i,m}))$$

Thus, the *M-step* reads

$$a_{i,j} = \frac{\sum_{r,t}\eta_{r,t}(i,j)}{\sum_j\sum_{r,t}\eta_{r,t}(i,j)}$$
$$s_{i,m} = \frac{\sum_{r,t}\gamma_{r,t}(i,m)}{\sum_m\sum_{r,t}\gamma_{r,t}(i,m)}$$
$$\mu_{i,m} = \frac{\sum_{r,t}\gamma_{r,t}(i,m)o_{r,t}}{\sum_{r,t}\gamma_{r,t}(i,m)}$$
$$U_{i,m} = \frac{\sum_{r,t}\gamma_{r,t}(i,m)(o_{r,t}-\hat{\mu}_{i,m})(o_{r,t}-\hat{\mu}_{i,m})^T}{\sum_{r,t}\gamma_{r,t}(i,m)}$$

where r indexes samples, and with

$$\eta_{r,t}(i,j) = \mathbb{E}_q[q_{r,t}^i, q_{r,t+1}^j] = \mathbb{P}_q[q_t^{(r)}=i, q_{t+1}^{(r)}=j]$$
$$\gamma_{r,t}(i,m) = \mathbb{E}_{q,c}[q_{r,t}^i c_{r,t}^m] = \mathbb{P}_{q,c}[q_t^{(r)}=i, c_t^{(r)}=m]$$

For a simpler step, the distribution are chosen deterministically equal to the mode, computed with Viterbi decoding.

B. *E-step*

The *M-step* have shown two quantities to be computed during the *E-step*: γ and η. Those are deduced from the choice of distribution for q and c. To minimize the relaxation gap, one should take $q\sim q|o$ and $c\sim c|o$. Let's recall useful factorizations for HMM. First

$$\begin{aligned} p(q_t, o_1,\dots,o_t) &= p(o_t|q_t)p(q_t,o_1,\dots,o_{t-1}) \\ &= p(o_t|q_t)\sum_{q_{t-1}} p(q_t,q_{t-1},o_1,\dots,o_{t-1}) \\ &= p(o_t|q_t)\sum_{q_{t-1}} p(q_t|q_{t-1})p(q_{t-1}|o_1,\dots,o_{t-1}) \end{aligned}$$

which allows fast recursive computation of $\alpha_t(q_t) = p(q_t,o_1,\dots,o_t)$. Same holds with $\beta_t(q_t) = p(o_{t+1},\dots,o_T|q_t)$ since

$$\begin{aligned} p(o_t,\dots,o_T|q_{t-1}) &= \sum_{q_t} p(q_t,o_t,\dots,o_T|q_{t-1}) \\ &= \sum_{q_t} p(q_t,o_t|q_{t-1})p(o_{t+1}\dots,o_T|q_t) \\ &= \sum_{q_t} p(o_t|q_t)p(q_t|q_{t-1})p(o_{t+1}\dots,o_T|q_t) \end{aligned}$$

Then, η is easily computable from

$$\begin{aligned} p(q_t,q_{t+1}|o_1,\dots,o_T) &\propto p(q_t,q_{t+1},o_1,\dots,o_T) \\ &= \alpha_t(q_t)\beta_{t+1}(q_{t+1})p(q_{t+1}|q_t)p(o_{t+1}|q_{t+1}) \end{aligned}$$

And γ is computed after factorization as

$$p(c_t,q_t|o) = p(q_t|o)p(c_t|o_t,q_t)$$

The first quantity is obtained as

$$p(q_t|o_1,\dots,o_T) \propto p(q_t,o_1,\dots,o_T) = \alpha_t(q_t)\beta_t(q_t)$$

And the second as a Gaussian mixture model

$$p(c_t|o_t,q_t) \propto s_{i,m}\mathcal{N}(o_t,\mu_{i,m},U_{i,m})$$

Note that the same holds for quantities $p(o_t|q_t)$, that appeared during the α and β recursions.

Case of several observations.: In practice, since we have several observation, we want to obtain $q|o^{(1)},\dots,o^{(T)}$. We didn't intend to look more in detail into calculations to derive this. We stayed with η_r and γ_r given under $q|o^{(r)}$.

C. *Viterbi decoding*

Let's now focus on the calculation of the mode. The same factorization ideas hold since $(\mathbb{R},\max,\times)$ is a semi-ring. This is capture by the so-called "Viterbi decoding" procedure. To obtain $\arg\max_q p(q|o)$, we derive after factorization of $\max_q p(q,o)$ the two recursive quantities. Forward, from $\xi_0=\delta_1$.

$$\xi_t(q_{t+1}) = \max_{q_t}\xi_{t-1}(q_t)p(o_t|q_t)p(q_{t+1}|q_t)$$
$$\xi_T = \max_{q_T}\xi_{T-1}p(o_T|q_T) \propto \max_q p(q|o)$$

And backward

$$q_T^* = \arg\max_{q_T} \xi_{T-1}(q_T) p(o_T|q_T)$$
$$q_t^* = \arg\max_{q_t} \xi_{t-1}(q_t) p(o_t|q_t) p(q_{t+1}^*|q_t)$$

Then, the mixture mode is approximated with

$$c_t = \arg\max_{m} p(o_t|q_t^*, c_t = m)$$

References

[1] M. Brand, N. Oliver, and A. Pentland. coupled hidden markov models for complex action recognition. *IEEE International Conference on Computer Vision and Pattern*, 1997.

[2] Z. Ghahramani and M. Jordan. factorial hidden markov models. *Kluwer Academic Publishers, Boston*, 1997.

[3] F.-J. Huang and T. Chen. real-time lip-synch face animation driven by human voice. *IEEE Workshop on Multimedia Signal Processing*, 1998.

[4] J. Luettin, G. Potamianos, and C. Neti. ”asynchronous stream modeling for large vocabulary audio-visual speech recognition”. *IEEE Int. Conf. Acoustics, Speech, Signal Processing*, 2001.

[5] A. Nefian, L. Liang, X. Pi, X. Liu, and K. Murphy. dynamic bayesian networks for audio-visual speech recognition. *EURASIP Journal on Applied Signal Processing*, 2002.

[6] L. Rabiner. a tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 1999.

[7] L. Rabiner and B-H. Juang. *Fundamentals of Speech Recognition*. Prentice-Hall, 1993.