Incremental learning of dynamical models of faces

Cyril Charron[1]
CharronC@cardiff.ac.uk
Yulia Hicks[1]
HicksYA@cardiff.ac.uk
Peter Hall[2]
pmh@cs.bath.ac.uk
Darren Cosker[2]
dpc@cs.bath.ac.uk

[1] School of Engineering
Cardiff University
Cardiff, UK

[2] Department of Computer Science
University of Bath
Bath, UK

Abstract

Active Appearance Models (AAM) are a useful and popular tool for modelling facial variations. They have been used in face tracking, recognition and synthesis applications. For modelling facial dynamics of speech, they have been used in conjunction with Hidden Markov Models (HMM). However, the high dimensionality of the training data and of the resulting AAMs leads to long learning time of HMMs and thus imposes serious limitations on their joint use.

Here, we propose a new method for learning HMMs of facial dynamics incrementally. Our algorithm is fully unsupervised and can be used for on-line learning as new data becomes available. Another important feature of our algorithm is the automatic choice of the number of states in the model. We show in experiments an improvement in learning speed of three orders of magnitude. Finally, we demonstrate the quality of the learned HMMs by generating video footage of a talking face.

1 Introduction and background

Active Appearance Models (AAM) [4] have been used widely to model dataset variations in the computer vision and, more recently, in the computer graphics applications. In some applications, they were used in conjunction with HMMs to represent the temporal dependencies in the datasets. For example, Cosker *et al.* [5] used AAMs together with HMMs to model facial variations during speech.

However, such data sets can be both of high dimensionality and contain a large number of samples, which leads to the following problems: 1) the number of samples in a data set must increase dramatically in relationship to the dimensionality of the samples to represent the data well; 2) the time complexity of the majority of HMM fitting algorithms grows rapidly with the number of states in the model; 3) the number of samples can be so large that they may exceed space resources; 4) once the models are learned, they typically cannot be changed to represent the additional data. A possible solution to the above problems would be an incremental approach to learning of HMM.

Incremental learning of HMM is an under-researched area. Continuous HMMs, which we consider in this article, are closely related to Gaussian Mixture Models (GMM). While substantial effort has been put into developing methods for incremental learning of GMMs [6, 10, 20], most of the accepted methods for learning GMM are still batch methods, which means that the GMM is learned from all data at once.

A standard approach to unsupervised batch learning is to use EM [7] to fit a sequence of GMMs, each with a specified number of components. The optimal model is selected from this candidate set using some penalty function, many of which exist including Akaike's Information Criterion [1], Minimum Description Length [15], Minimum Message Length [19], and Bayesian approaches [16]. An alternative approach is to integrate choosing the number of components with fitting as proposed by Figueiredo and Jain in [9], which produces the best results for batch methods. However, all of these suffer from the problems mentioned earlier.

Recursive algorithms are an alternative to batch methods. In [17], Titterington presented a recursive algorithm for updating an existing GMM when new data samples become available. Later, Zivkovic and van der Heijden [20] added a selection mechanism to a similar recursive algorithm and applied it to online learning of an adaptive model of background in [21]. However, whilst the selection heuristic they use is computationally efficient, the theoretical support for the finally selected number of components is questionable. In this work, we propose an alternative solution to incremental update of GMMs based on the incremental clustering of mixture model components, as opposed to the incremental clustering of points as in the above algorithms. Our approach is computationally efficient as it works with components rather than points; it also includes a selection mechanism relying on the MML criterion and thus is theoretically sound.

The clustering of mixture models was first presented in the work of Vasconcelos and Lippman [18] as a solution for hierarchical mixture learning. More recently, Goldberger and Roweis [10] and Davis and Dhillon [6] proposed alternative approaches to clustering of mixture models. However none of the above work established a link between clustering of mixture models and incremental learning algorithms.

Our approach to incremental update of GMMs is easily extended to incremental update of HMMs as we show in section 2.3. Our algorithm is fully unsupervised and is both efficient in terms of memory and fast. Due to the built-in selection mechanism, the updated states describe the data optimally in the Minimum Message Length (MML) sense, thus being compact. We show in experiments an improvement in learning speed of three orders of magnitude, while producing comparable likelihood scores at a similar level of complexity. Finally, we demonstrate the quality of the learned HMMs by generating video footage of a talking face.

2 Learning dynamical models incrementally

There are two major stages in our algorithm. In the first stage, we update the observation distribution of the existing HMM by merging it with that of the incoming HMM. This is done using a new principled Expectation-Maximisation (EM) procedure which does not require any recourse to the training data, but simply relies on the state descriptions. Thus, this stage is both efficient in terms of memory and fast. Due to the built-in selection mechanism, the updated states describe the data optimally in the Minimum Message Length (MML) sense, thus being compact. In the second stage of our algorithm, we use an approach proposed by Hicks and Hall [13] to update the transition matrix and the initial state priors of the new HMM, which is done in time linearly dependent on the number of states. In the following sections, these stages are described in detail.

2.1 Updating the observation distributions

In this part, we explain the process of updating an existing HMM observation distribution (expressed by a GMM) with the information encoded by a new incoming HMM (GMM part of it). Thus, we estimate the parameters of a new GMM from the descriptions of two existing GMMs. This new GMM should satisfy two constraints: 1) it has to describe the data well, 2) it should avoid over- or under-fitting the data to offer good generalization and insights on the phenomenon to the user. We do that using a new EM procedure which does not require any recourse to the training data, but simply relies on the state descriptions.

As in the classical EM procedure, we want to find the parameters of our new model θ_{new} maximising the likelihood of the data set $\mathbf{X}$. We can use the previously learned models represented by their concatenation θ_{old} to split the data set into independent blocks $\{X_i\}_{i=1:C^{old}}$ (as previously done by Vasconcelos and Lippman [18]). The likelihood of the data set $\mathbf{X}$ can thus be split in the product of data block likelihood (Eq. (1)).

$$p(\mathbf{X}|\theta_{new}) = \prod_{i=1}^{C_{old}} p(\mathbf{X}_i|\theta_{new}) \tag{1}$$

We assume that all the samples in a given block will be attributed to a single component in our model. Namely, we attribute a hidden label z_{ij} that takes on the value 1 if the block X_i is encompassed by the $j-$th component θ_j^{new} of the new model and the value 0 otherwise. Similarly to the labels in the classical EM algorithm, this is a binary vector which has only one non-zero component.

$$p(\mathbf{X}_i|\theta_{new}) = \sum_{j=1}^{C_{new}} \alpha_j p(\mathbf{X}_i|z_{ij}=1,\theta_{new}) = \sum_{j=1}^{C_{new}} \prod_{m=1}^{n} \alpha_n p(\mathbf{x}_i^{(m)}|z_{ij}=1,\theta_{new}) \tag{2}$$

The α_j in Eq. (2) are the priors of the components in the new model. Using equations (1) and (2), we can express the likelihood of the complete data set $\{X,Z\}$, i.e. the samples and their labels, using Eq. (3). This function is equivalent to the classical Q function [8].

$$p(\mathbf{X},\mathbf{Z}|\theta_{new}) = \prod_{i=1}^{C_{old}} \prod_{j=1}^{C_{new}} [\alpha_j p(\mathbf{X_i}|z_{ij}=1,\theta_{new})]^{z_{ij}} \tag{3}$$

maximising this function is equivalent to maximising the likelihood of the data. We proceed in two steps: 1) we first evaluate the expected value of the labels knowing the sample blocks and an estimate of the model parameters (similar to the classical E-step); 2) we update these estimates to maximise the Q function, knowing the sample blocks and the expected value of the labels (similar to the M-step). Note that instead of working with the samples, we work with the blocks of samples here, which is similar to [18].

The expected value h_{ij} of the $i-$th block label is given in Eq. (4).

$$h_{ij} = E[z_{ij}|\mathbf{X}_i,\theta_{new}] = p(z_{ij}=1|\mathbf{X_i},\theta_{new}) = \frac{p(\mathbf{X}_i|z_{ij}=1,\theta_{new})\alpha_j}{\sum_l p(\mathbf{X}_i|z_{il}=1,\theta_{new})\alpha_l} \tag{4}$$

We do not have access to the samples in $\mathbf{X}_i$, so we cannot evaluate directly $P(\mathbf{X}_i|z_{ij},\theta_{new})$. Instead, we evaluate its expected value. The points in $\mathbf{X}_i$ are distributed according to the i-th component of the existing model. Thus, the expected probability of a single sample from $\mathbf{X}_i$, knowing the current estimate $\hat{\theta}_j^{new}$ of the j-th component of the new model, is given by the integral in Eq. (5). Since there are $M_i = \alpha_i^{old} \times N$ samples in $\mathbf{X}_i$, this expression has to be raised to the power M_i.

$$E\left[P(X_i|z_{ij},\theta_{new})\right] = \left(\int p(x|\theta_i^{old})p(x|\hat{\theta}_j^{new})\right)^{M_i} \tag{5}$$

In the case of Gaussian distributions, the integral in Eq. (5) is the Bhattacharyya distance [2] between the $i-$th component of the old model and the $j-$th component of the new model, respectively parametrised by their priors $\alpha_{i,j}$, their means $\mu_{i,j}$ and their covariances $C_{i,j}$. Here, equations (5-7) are derived using a different approach from Vasconcelos and Lippman [18], *i.e.* incremental learning. This leads to an interesting difference in Eq.(6), which clearly contains a distance measure between two different distributions. Thus, this equation establishes a link between label expectations in probabilistic EM approaches for GMM learning [9, 18] and the distance used in graph based methods for clustering GMMs [6, 10, 12].

Replacing $P(\mathbf{X}_i|z_{ij},\theta_{new})$ by its expected value in Eq. (4), yields the expression of h_{ij} in Eq. (6).

$$\begin{aligned} h_{ij} &= \frac{\exp\left(-M_i\rho_{ij}\right)\alpha_j}{\sum_l \exp\left(-M_i\rho_{il}\right)\alpha_l} \\ \text{with}\quad \rho_{ij} &= \frac{1}{8}(\mu_i^{old}-\mu_j^{new})^T(C_i^{old}+C_j^{new})^{-1}(\mu_i^{old}-\mu_j^{new}) + \frac{1}{2}\log\frac{|C_i^{old}+C_j^{new}|}{|C_i^{old}|^{\frac{1}{2}}|C_j^{new}|^{\frac{1}{2}}}, \end{aligned} \tag{6}$$

where ρ_{ij} is the Bhattacharyya distance between two Gaussian distributions. We can now proceed with the update of the parameters of $\hat{\theta}^{new}$ to maximise the Q-function in Eq. (3) with the constraint $\sum_j \alpha_j = 1$ (this is equivalent to the M-step in the classical EM algorithm). This is done using the Lagrange multipliers, which results in the updated parameters given in Eq. (7).

$$\left\{\begin{array}{lcl} \hat{\alpha}_j & = & \sum_{i=1}^{K^{old}} h_{ij}\alpha_i \\ \hat{\mu}_j & = & \frac{1}{\hat{\alpha}_i}\sum_{i=1}^{K^{old}} h_{ij}\alpha_i\mu_i \\ \hat{C}_j & = & \frac{1}{\hat{\alpha}_j}\left(\sum_{i=1}^{K^{old}} h_{ij}\alpha_i(C_i+\mu_i\mu_i^T)\right) - \hat{\mu}_j\hat{\mu}_j^T \end{array}\right. \tag{7}$$

Finally, note that the new model need not have the same number of parameters as the old one. This is discussed in the next section.

2.2 Choosing the number of components

Up to now, we let the number of components in the new model to be equal to an arbitrary value K^{new}. It is obvious that the initial model θ^{old} will be redundant (by construction) and thus will over-fit the data. Our model should then have fewer components. Choosing the appropriate number of components can also reveal underlying structures in the data.

The EM algorithm maximises the likelihood of the data. However, as pointed out by Figueiredo and Jain [9], it is always possible to increase the likelihood by adding more components to the model. To solve this problem, we could introduce a criterion aiming at maximising the log likelihood of the data while penalising complex solutions. Such criteria are the Akaike Information Criterion [1], the Minimum Description Length [15], the Minimum Message Length [19], and the Bayesian criterion proposed by Roberts [16].

Equivalently, Figueiredo and Jain [9] use a cost function (Eq. (8)) that has to be minimised. We use their cost function with the term referring to the samples replaced by their expectation value as in Eq. (4). This yields a new cost function (Eq. (9)) which no longer depends on the samples but only on the model parameters. Briefly, the first sum is the optimum length of the model description, while the last term is the code length of the data (see [9] for more details). Γ is related to the number of parameters in the model.

$$\mathscr{L}(\theta,\mathbf{X}) = \Gamma \sum_{j=1}^{K^{new}} \log(N\alpha_j) + \frac{K^{new}}{2}\left(1+\log\frac{N}{12}\right) - \log P(X|\theta^{new}) \tag{8}$$

$$\mathscr{L}(\theta,\mathbf{X}) \sim \Gamma \sum_{j=1}^{K^{new}} \log(N\alpha_j) + \frac{K^{new}}{2}\left(1+\log\frac{N}{12}\right) - \sum_i \sum_j M_i E_{\theta_{old},i}[\log p(\mathbf{x}|z_{ij}=1,\theta_{new})] \tag{9}$$

This cost function is equivalent to choosing Dirichlet priors for the α_i's. Dirichlet priors have been used by Figueiredo and Jain [9] to discard redundant components in the GMM. We introduce them into our algorithm by taking the MAP estimate instead ML estimate of the α_i's (Eq. (7)) and thus obtain the modified M-step given in Eq. (10). This update suppresses the components having too little support from the data.

$$\hat{\alpha}_j = \frac{\max\left\{0, \left(\sum_{i=1}^{K^{old}} h_i j\right) - \Gamma\right\}}{\sum_{l=1}^{K^{new}} \max\left\{0, \left(\sum_{i=1}^{K^{old}} h_i l\right) - \Gamma\right\}} \tag{10}$$

Below, we summarise the new EM procedure including automatic choice of the number of components. We initialise our procedure by over-fitting the probability density function, which is modelled by the concatenation of the components from the HMM to be updated and the new incoming HMM. The expectation step (Eq. (6)) and the maximisation step (Eq. (7) and (10)) are iteratively applied. The suppression mechanism implemented in Eq. (10) iteratively deletes weak components. Note that when the cost function does not evolve anymore, the weakest component of the model is deleted. This allows the procedure not to get stuck in local minima. Eventually the number of components reaches a chosen threshold and the model having the minimum cost is returned.

2.3 Merging HMMs

In [13], Hicks and Hall proposed a method to merge HMMs based on the merging of their observation distributions expressed by GMMs as in the present paper). The key idea is to use the linear combination in Eq. (7) mapping the old observation distributions to the new ones to obtain the state transition matrix and the state priors of the new HMM. Their method is summarised below for completeness.

Let us define two HMMs $\lambda_1 = (A^1, B^1, \pi^1)$ and $\lambda_2 = (A^2, B^2, \pi^2)$ respectively described by their initial state probability vectors π^i, their state transition probability matrices A^i and their distributions representing the states B^i. The first HMM, λ_1, has M states, whereas the second one has N. Let $\lambda_3 = (A^3, B^3, \pi^3)$ denote the result of their addition, consisting in $K \leq M+N$ states (K being defined during the GMM merging process).

To obtain the new transition matrix, we start with concatenating the transitions matrices A^1 and A^2 into a single matrix A^c of size $n = M + N$ (Eq. (11)).

$$\begin{aligned} A^c &= \begin{pmatrix} a_{1,1} & \dots & a_{1,M} & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{M,1} & \dots & a_{M,M} & 0 & \dots & 0 \\ 0 & \dots & 0 & a_{M+1,M+1} & \dots & a_{M+1,M+N} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & a_{M+N,M+1} & \dots & a_{M+N,M+N} \end{pmatrix} \\ \text{where } a_{ij} &= \begin{cases} \{A^1\}_{ij} & \text{if } i \leq M \text{ and } j \leq M \\ \{A^2\}_{ij} & \text{if } i > M \text{ and } j > M \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{11}$$

The elements of the probability transition matrix A^3 are obtained using Eq. (12), where α_i denotes the component priors, $a_{i,j}$ the elements of the matrix A^c and h_{ij} the linear combination coefficient.

$$\{A^3\}_{ij} = \frac{\sum_{l=1}^{n} \sum_{k=1}^{n} \alpha_k h_{ki} h_{lj} a_{kl}}{\sum_{j=1}^{K} \sum_{l=1}^{n} \sum_{k=1}^{n} \alpha_k h_{ki} h_{lj} a_{kl}} \tag{12}$$

The initial state probability vector π_3 can be obtained in a similar way. We first concatenate the two initial state probability vectors π_1 and π_2 into a single vector π_c, and define the elements of π_3 according to Eq. (13).

$$\pi_i^3 = \frac{\sum_{k=1}^{n} h_{ki} \pi_k^c}{\sum_{i=1}^{K} \sum_{k=1}^{n} h_{ki} \pi_k^c} \tag{13}$$

3 Experiments on synthetic data sets

In this section, we present quantitative assessment of our algorithm on a widely used data set: the shrinking spiral. Data samples are generated according to Eq. (14), where $t \in [0, 4\pi]$, and n has Gaussian distribution $\mathcal{N}(0, I)$. An interesting property of this dataset is that it forms a one dimensional manifold embedded in an higher dimensional space $\mathbb{R}^3$, which is similar to the trajectories of the faces we will analyse in section 4.

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} (13 - 0.5t_i)\cos t_i \\ (0.5t_i - 13)\sin t_i \\ t_i \end{bmatrix} + \begin{bmatrix} n_x^i \\ n_y^i \\ n_z^i \end{bmatrix} \tag{14}$$

In this section, we want to mimic online incremental learning of a GMM describing the above dataset. We generate the main test set consisting of $1,250$ points that we split (over 20 trials) into a randomly chosen number, ranging from two to ten, of smaller subsets. A standard batch method developed by Figueiredo and Jain [9] with a built-in selection mechanism is applied to the main set resulting in a model which constitutes our baseline. The same batch method is also applied to every subset producing local models (as opposed to the global model learned on the whole set).

To start our incremental learning procedure, we concatenate the descriptions of the first two local models and use the result as the input of our procedure. We then concatenate the result of our method to the next local model and apply our merging procedure again. This

	batch mixtures	concatenated mixtures	incremental mixtures
Nb of components	17 ± 0	31 ± 0	$17 \pm 0.$
log likelihood	-7.77 ± 0.03	-7.78 ± 0.02	-7.75 ± 0.03
RBC ($\times 10^4$)	4.56 ± 0.03	4.64 ± 0.02	4.55 ± 0.01
Time (s)	53 ± 2	4.2 ± 0.7	15 ± 3

Table 1: Comparison of the batch and incremental methods for learning observation distribution. The concatenation is given for comparison.

is repeated until all local models have been merged. As a quick check, we also look at the effect of just concatenating the local models.

Table 1 presents different measurements obtained with the three different models (batch, our incremental and the simple concatenation). The measurements are averaged over 20 trials. One of the important characteristic of our method is its automatic selection mechanism, thus we present the number of components of the models. As can be seen, our incremental method and the batch method produce similar models in terms of complexity. The models resulting from the concatenation overfit in the MML sense the data as they have considerably more components.

Next we present the likelihoods of the different models. It is interesting to see that the concatenated models, which constitute the input of our algorithm, tend to produce worse results. Our method produces slightly better results than the batch method but results are not significant. Our results replicate those presented in [20].

We look at the Bayesian criterion proposed by Roberts (RBC) [16], which takes into account the likelihood and penalises the complexity of the models. As expected, the concatenated model has the worst RBC, whilst our method is comparable with the batch method.

Finally, we look at the computational times of the different methods and this is where our incremental method shines. Please note that all methods were evaluated with the same Matlab code on a 2.33GHz machine (the batch algorithm is from the publicly available GMMBayes toolbox and our incremental code is based on it). The batch method takes 53 sec to learn the $1,250$ samples of the main set, while our method (including the concatenation times) takes 15 sec. However, this improved efficiency is below what the theory would predict. The main drawback of our implementation lies in the evaluation of the label expectation h_{ij} which requires to compute the Bhattacharyya distance (which is more complex to evaluate than the Mahalanobis distance used in the batch method) and there is still some room for improving the implementation here (see Comaniciu *et al.* [3]).

4 Learning dynamical models of faces

In this section, we apply our method to learning HMMs modelling the dynamics of a video realistic human face. In [5], Active Appearance Models (AAM) were trained on videos of talking people in order to produce new realistic synthetic videos for computer graphics applications. A continuous HMM was learned from the same training data to model their dynamics.

The dimensionality of the training data and the number of samples necessary to learn such models is usually very large. Thus, in the original paper [5], incremental methods [11] were used for constructing the AAM to circumvent memory issues.

Nonetheless, in [5], the HMMs are still learned on the whole dataset projected onto the AAM eigenspace. This has three major drawbacks: 1) the Baum-Welch algorithm used

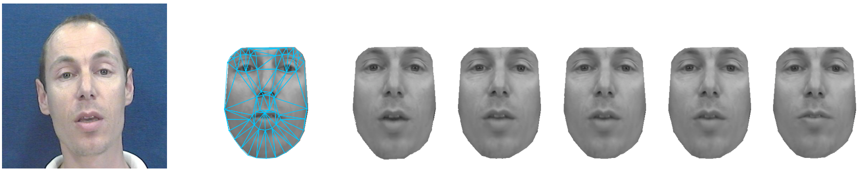


Figure 1: A frame from the video sequence (left) and the mesh and texture luminance extracted from it. Next, five synthetised frames 340 to 390 (10 frames steps).

to learn the HMM is of complexity $\mathcal{O}(Tk^2)$ [14], where T is the length of the sequence (number of samples) and k is the number of states in the HMM; 2) would a new unseen sequence of data become available, the whole procedure would have to be repeated on the union of both the original and the new set, wasting the effort produced on the previous set; 3) the previous data may not be available anymore (limitation of space). Moreover, in [5], the number of components of the HMM is defined after a careful study of the outcomes, too few components making the synthetic video look jerky.

In the following section we shall apply the incremental procedure presented in this article to learn incrementally an HMM of facial dynamics on a similar dataset.

We use the same database as in [5], provided by Cosker *et al.*, consisting in $4,173$ frames (720×576-pixels) from a video of a person telling a story (Fig-1). In each frame, 63 feature points – located at strategic facial positions (e.g, nostrils, eye contour, lips borders, etc.) – are tracked. These points model the shape of the face and allow to extract a normalised texture (luminance information) of the pixels located inside the convex hull defined by the feature points. Thus, each video frame is represented by a shape-texture vector consisting of around $50,000$ elements, making the whole data set very large. It was split into 28 subsets of 150 vectors each (this number being governed by memory limitations on our computer). The AAM model parameters were learned according to the incremental scheme described in [5] and constituted the input to the HMM learning.

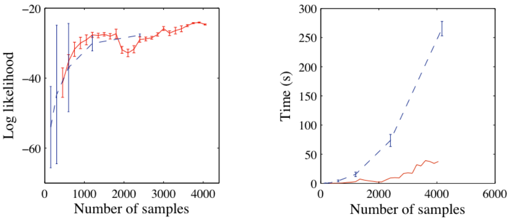
4.1 Experiment

We ran several trials of the standard HMM learning algorithm for different sizes of training sets, each time assessing the likelihood of the produced HMM, as well as the computational time. For our method, the process consist of two phases: 1) learning a local HMM on the arriving data, and 2) merging these model with the previous ones. The time to consider in the evaluation is then the sum of steps 1) and 2). The likelihoods of the GMMs are evaluated on random points of the testing set unseen during the training phase. A similar evaluation is done for HMMs.

Synthetic videos were produced from the learned HMM to assess visually their quality. We were looking at the 'smoothness' and the variety of dynamical facial expression as well as the details of the face (high frequency information).

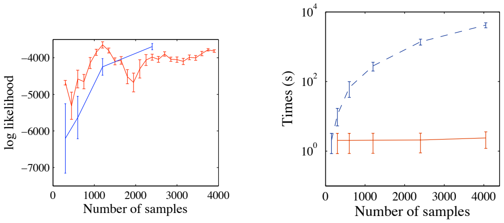
4.2 Discussion and results

An example of a typical video produced by our algorithm can be found in the supplementary material and some frames are shown in Fig-1. The dynamics of the synthetic face are smooth, with no jerky changes, and successfully produce an impression of a person speaking. The



(a) (b) (c)

Figure 2: a) learning time of the GMM obtained with **FJ** algorithm (dashed line) and with our method (plain line), with respect to the size of the training set. b) Log likelihood of the respective methods and c) number of components in the GMM.



(a) (b)

Figure 3: a) Log likelihood of the HMM produced by the Baum-Welch procedure and log likelihood for the HMM obtained using our method (plain line). b) Computation time of the HMM learning. Please, note the log scale.

texture contains fine details and we did not have to use any sharpen filter to enhance them. The animation is comparable to what Cosker *et al.* obtained with the same data set (with exclusion of the results produced by hierarchical models [5]).

The incremental update of the GMM requires considerably less time than the standard batch method (Fig-2). It is very important to note that the GMMs produced by both methods are equivalent in terms of their likelihood, and that it is not due to our model using more components, thus showing that our selection criterion is working properly.

Finally, the most striking results come from the incremental update of the HMMs. The evolution of the HMM likelihood for both methods – Baum-Welch and ours – with respect to the size of the training set is shown in Fig-3. The quality of both models improves as the number of samples increases. The quality of the incremental model is comparable with that produced by the standard batch method. The computation times of the incremental method are up to three orders of magnitude faster than the Baum-Welch procedure.

5 Conclusions

In this article, we presented a new unsupervised method for learning HMM incrementally from the descriptions of the existing HMMs using a new principled EM procedure. The method has a built-in selection mechanism of the number of components based on an MML criterion. Moreover, the whole procedure does not require any direct access to the training data and only relies on the existing models parameters.

The above listed properties of our algorithm make it uniquely efficient it terms of memory and computational time as we show in the synthetic and real world experiments.

The models produced by our incremental algorithm are of comparable quality to those produced by a standard batch method with similar likelihood scores at similar levels of complexity. We also demonstrated the quality of the learned HMMs by generating a video footage of a realistically looking talking face from a learned HMM.

Acknowledgements

This work has been funded by EPSRC grant EP/E017576/1.

References

[1] Information theory and an extension of the maximum likelihood principle. *Second International Symposium on Information Theory*, pages 267–281, 1973.

[2] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by probability distributions. *Bull. Calcutta Math. Soc.*, 35:99–109, 1943.

[3] D. Comaniciu, P. Meer, and D. Tyler. Dissimilarity computation through low rank corrections. *Pattern Recognition Letters*, 24(1-3):227–236, January 2003.

[4] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE Pattern Analysis and Machine Intelligence*, 23(6):681–685, 2001.

[5] D. Cosker, D. Marshall, P. Rosin, and Y.A. Hicks. Speech driven facial animation using a hidden markov co-articulation model. *Proc. of IEEE International Conference on Pattern Recognition (ICPR)*, 1:128–131, 2004.

[6] J.V. Davis and I. Dhillon. Differential entropic clustering of multivariate gaussians. *Advances in Neural Information Processing Systems*, 19:337–344, 2007.

[7] A.P. Dempster, N.M. Laird, and DB Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[8] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley & Son, 2nd edition edition, 2001.

[9] M. A. F. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on pattern analysis and machine intelligence*, 24(3):381–396, 2002.

[10] J. Goldberger and S. Roweis. Hierarchical clustering of a mixture model. *Advances in Neural Information Processing Systems*, 17:505–512, 2005.

[11] P. Hall, D. Marshall, and R. Martin. Merging and splitting eigenspace models. *IEEE Transactions on pattern analysis and machine intelligence*, 22(9):1042–1049, 2000.

[12] P. Hall, Y. Hicks, and T. Robinson. A method to add gaussian mixture models. Technical report, Department of Computer Science, University of Bath, 2005.

[13] Y.A. Hicks, P.M. Hall, and A.D. Marshall. A method to add hidden markov models with application to learning articulated motion. *British Machine Vision Conference*, 2003.

[14] L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, 3(1):4–16, January 86.

[15] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[16] S. J. Roberts, D. Husmeier, I. Rezek, and W. Penny. Bayesian approaches to gaussian mixture modeling. *IEEE Transactions on Pattern Analysis and Machine Intellingence*, 20(11):1133–1142, 1998.

[17] D. M. Titterington. Recursive parameter estimation using incomplete data. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, 46:257–267, 1984.

[18] N. Vasconcelos and A. Lippman. Learning mixture hierarchies. *Advances in Neural Information Processing Systems*, 11:606–612, 1999.

[19] C. Wallace and D. Boulton. An information measure for classification. *Computer Journal*, 11:195–209, 1968.

[20] Z. Zivkovic and F. van der Heijden. Recursive unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):651–656, Jan 2004.

[21] Z. Zivkovic and F. van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27(7):773–780, 2006.