CONVOLUTIONAL DENSITY ESTIMATION IN HIDDEN MARKOV MODELS FOR SPEECH RECOGNITION

Spyros Matsoukas, George Zavaliagkos

BBN Technologies, GTE Internetworking
70 Fawcett Street, Cambridge, MA 02138
smatsouk@bbn.com

ABSTRACT

In continuous density Hidden Markov Models (HMMs) for speech recognition, the probability density function (pdf) for each state is usually expressed as a mixture of Gaussians. In this paper, we present a model in which the pdf is expressed as the convolution of two densities. We focus on the special case where one of the convolved densities is a M-Gaussian mixture, and the other is a mixture of N impulses. We present the reestimation formulae for the parameters of the $M \times N$ convolutional model, and suggest two ways for initializing them, the residual K-Means approach, and the deconvolution from a standard HMM with MN Gaussians per state using a genetic algorithm to search for the optimal assignment of Gaussians. Both methods result in a compact representation that requires only $\mathcal{O}(M+N)$ storage space for the model parameters, and $\mathcal{O}(MN)$ time for training and decoding. We explain how the decoding time can be reduced to $\mathcal{O}(M+kN)$, where $k < M$. Finally, results are shown on the 1996 Hub-4 Development test, demonstrating that a 32 $\times$ 2 convolutional model can achieve performance comparable to that of a standard 64-Gaussian per state model.

1. INTRODUCTION

In most speech recognition systems based on the traditional HMM, a set of observations $X = (x_1, \dots, x_T)$ is assumed to be generated by a sequence of HMM states, where the pdf of each state is a Gaussian mixture of the form

$$p(x) = \sum_{k=1}^{K} w_k \mathcal{N}(x; \mu_k, C_k)$$

In this paper, we use a more structured representation for the pdf; we assume that each observation x_t is generated by the convolution of two densities $p_y(y)$ and $p_b(b)$, where $p_y(y)$ is a mixture of N impulses

$$p_y(y) = \sum_{j=1}^{N} q_j \delta(y - m_j)$$

and $p_b(b)$ is a Gaussian mixture of M components

$$p_b(b) = \sum_{i=1}^{M} p_i \mathcal{N}(b; \mu_i, C_i)$$

Therefore,

$$\begin{aligned} p(x_t) &= \int p_b(b_t) p_y(x_t - b_t) db_t \\ &= \sum_{i=1}^{M} \sum_{j=1}^{N} p_i q_j \mathcal{N}(x_t; \mu_i + m_j, C_i) \end{aligned}$$

We can see that in this simple model the likelihood of each observation x_t can be computed trivially as with any other Gaussian mixture density. However, the joint reestimation of the parameters μ_i, m_j and C_i is more complicated than the standard case and requires the solution of a least squares problem.

In the following sections, we present the implementation details of the convolutional model. Section 2 shows the solution of the least squares problem for the reestimation of the parameters of the model, both in forward-backward training and in K-Means initialization. In section 3 we present two different ways to initialize the parameters of the convolutional model, the residual K-Means, and the deconvolution from an MN-bin HMM[1]. Section 4 describes the storage requirements of the convolutional model, and presents a method for reducing the likelihood computation during recognition, by taking advantage of the structure in the convolutional model. Experimental results on the 1996 Hub-4 development test are shown in section 5. Finally, we conclude with a summary and suggestions for future research in section 6.

2. PARAMETER REESTIMATION

In this section we present the reestimation formulae for the case where the density for each state $s = 1, \dots, S$ is the convolution of an M-component Gaussian mixture with an N-component impulse mixture. We assume that our observations are n-dimensional feature vectors.

2.1. Forward-Backward Training

In forward-backward training we seek to maximize the value of the auxiliary function

$$\mathcal{Q} = \sum_{s=1}^{S} \sum_{t=1}^{T} \sum_{i=1}^{M} \sum_{j=1}^{N} \gamma_s^{ij}(t) \mathcal{Q}_s^{ij}(t) \tag{1}$$

where

$$\mathcal{Q}_s^{ij}(t) = b_s^{ij} - \frac{1}{2}\mathcal{L}_s^{ij}(t) \tag{2}$$

$$b_s^{ij} = \ln p_{si} + \ln q_{sj} - \frac{n}{2}\ln(2\pi) \tag{3}$$

[1]In what follows, we will use the term K-bin model to describe an HMM with K Gaussians per state.

$$\begin{aligned}\mathcal{L}_s^{ij}(t) &= \ln(|C_{si}|) + &(4)\\ &\quad (x_t - \mu_{si} - m_{sj})' C_{si}^{-1} (x_t - \mu_{si} - m_{sj}) &(5)\end{aligned}$$

$\gamma_s^{ij}(t)$ is the probability that the observation x_t is generated by the cooperation of the ith Gaussian and jth impulse in state s; C_{si} is the covariance matrix of the ith Gaussian component in state s; μ_{si} and m_{sj} are the mean vectors of the ith Gaussian and jth impulse in state s, respectively.

By differentiating Q with respect to the parameters μ_{si} and m_{sj} we obtain:

$$\mu_{si} = \frac{\sum_{j=1}^{N}\sum_{t=1}^{T}\gamma_s^{ij}(t)(x_t - m_{sj})}{\sum_{j=1}^{N}\sum_{t=1}^{T}\gamma_s^{ij}(t)} \quad (6)$$

$$m_{sj} = \frac{\sum_{i=1}^{M} C_{si}^{-1}\sum_{t=1}^{T}\gamma_s^{ij}(t)(x_t - \mu_{si})}{\sum_{i=1}^{M} C_{si}^{-1}\sum_{t=1}^{T}\gamma_s^{ij}(t)} \quad (7)$$

By differentiating Q with respect to $z_s^{ij} = \mu_{si} + m_{sj}$, and C_{si}, we obtain:

$$z_s^{ij} = \frac{\sum_{t=1}^{T}\gamma_s^{ij}(t)x_t}{\sum_{t=1}^{T}\gamma_s^{ij}(t)} \quad (8)$$

$$C_{si} = \frac{\sum_{t=1}^{T}\gamma_s^{ij}(t)(x_t - z_s^{ij})(x_t - z_s^{ij})'}{\sum_{t=1}^{T}\gamma_s^{ij}(t)} \quad (9)$$

Similarly, the reestimation formulae for the mixture weights are:

$$p_{si} = \frac{\sum_{j=1}^{N}\sum_{t=1}^{T}\gamma_s^{ij}(t)}{\sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{t=1}^{T}\gamma_s^{ij}(t)} \quad (10)$$

$$q_{sj} = \frac{\sum_{i=1}^{M}\sum_{t=1}^{T}\gamma_s^{ij}(t)}{\sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{t=1}^{T}\gamma_s^{ij}(t)} \quad (11)$$

As we can see in equations 6 and 7, the solution for μ_{si} depends on m_{sj} and vice versa. If we assume diagonal covariance matrices C_{si}, then we can jointly reestimate the means in each dimension $d = 1, \ldots, n$ independently, by solving the following system:

$$\begin{bmatrix} D_s & F_s \\ F_s'\Lambda_s^d & G_s^d \end{bmatrix}\begin{bmatrix} \mu_s^d \\ m_s^d \end{bmatrix} = \begin{bmatrix} \alpha_s^d \\ \beta_s^d \end{bmatrix} \quad (12)$$

where

$$\begin{aligned}
F_s(i,j) &= \sum_{t=1}^{T}\gamma_s^{ij}(t)\\
D_s(i,i) &= \sum_{j=1}^{N} F_s(i,j), \quad D_s(i,j) = 0 \;\; i \neq j\\
\Lambda_s^d(i,i) &= C_{si}^{-1}(d,d), \quad \Lambda_s^d(i,j) = 0 \;\; i \neq j\\
G_s^d(j,j) &= \sum_{i=1}^{M}(F_s'\Lambda_s^d)(j,i), \quad G_s^d(j,i) = 0 \;\; i \neq j\\
\mu_s^d &= [\mu_{s1}(d) \cdots \mu_{sM}(d)]'\\
m_s^d &= [m_{s1}(d) \cdots m_{sN}(d)]'\\
\alpha_s^d &= [\alpha_{s1}(d) \cdots \alpha_{sM}(d)]'\\
\alpha_{si} &= \sum_{j=1}^{N}\sum_{t=1}^{T}\gamma_s^{ij}(t)x_t\\
\beta_s^d &= [\beta_{s1}(d) \cdots \beta_{sN}(d)]'\\
\beta_{sj} &= \sum_{i=1}^{M} C_{si}^{-1}\sum_{t=1}^{T}\gamma_s^{ij}(t)x_t
\end{aligned}$$

Note that the system in Eq. 12 has infinite number of solutions, since for every pair (μ_s^d, m_s^d) that satisfies Eq. 12, the pair $(\mu_s^d + \epsilon, m_s^d - \epsilon)$, $\epsilon > 0$, is also a solution to the system. Thus, we first find the minimum norm solution for m_s^d, using Singular Value Decomposition, as follows:

$$m_s^d = \left(G_s^d - F_s'\Lambda_s^d D_s^{-1} F_s\right)^{+}\left(\beta_s^d - F_s'\Lambda_s^d D_s^{-1}\alpha_s^d\right) \quad (13)$$

and then, we solve for μ_s^d using the formula

$$\mu_s^d = D_s^{-1}\left(\alpha_s^d - F_s m_s^d\right) \quad (14)$$

This solution has two advantages over the direct SVD solution from Eq. 12: First, in Eq. 13 we need to compute the pseudo-inverse of a $N \times N$ matrix, while direct SVD on Eq. 12 requires operations on an $(M + N) \times (M + N)$ matrix. If $N < M$, which is usually the case in our experiments, then Eq. 13 results in a faster solution. Second, in the special case where the impulse mixture consists of a single mean, i. e. $N = 1$, then Eq. 13 returns 0 into m_s^d, and the solution for μ_s^d is identical to the standard M-component Gaussian mixture HMM.

2.2. K-Means

In K-Means [1], we reestimate the means by solving a linear system similar to Eq. 12:

$$\begin{bmatrix} D_s & F_s \\ F_s' & G_s \end{bmatrix}\begin{bmatrix} \mu_s^d \\ m_s^d \end{bmatrix} = \begin{bmatrix} \alpha_s^d \\ \beta_s^d \end{bmatrix} \quad (15)$$

where

$$\begin{aligned}
G_s(j,j) &= \sum_{i=1}^{M} F_s(i,j), \quad G_s(j,i) = 0 \;\; i \neq j\\
\beta_{sj} &= \sum_{i=1}^{M}\sum_{t=1}^{T}\gamma_s^{ij}(t)x_t
\end{aligned}$$

As we can see, the composite matrix in Eq. 15 is independent of the dimension d, and therefore the K-Means reestimation requires only one matrix inversion for all n dimensions.

The variances are reestimated in the last iteration of K-Means, by keeping the means fixed and calculating the variance of the data around them, in the usual way.

2.3. Approximate solution in forward-backward

Notice that in the forward-backward reestimation, although we estimate $M + N$ mean vectors and M variances per state, we still need to store $\mathcal{O}(MN)$ statistics, in order to solve the linear system in Eq. 12. This may result in excessive disk storage requirements and I/O overhead when MN is large. To avoid this problem, we have implemented an approximate solution to the forward-backward reestimation, that requires $\mathcal{O}(M + N)$ space.

We perform a total of four passes of forward-backward training: in the first and third passes, we keep the impulse means m_{sj} fixed, and we reestimate the Gaussian means μ_{si} and variances C_{si}. In the second and fourth passes, we keep the variances fixed, and we reestimate both the μ_{si} and m_{sj} means. This approximation achieves the same accuracy as the exact solution, with significantly smaller storage and I/O requirements.

3. MODEL INITIALIZATION

We have found that the recognition accuracy of the convolutional models depends heavily on the initialization of the K-Means. In this section, we propose two methods for initializing the model parameters, and we compare their performance in section 5.

3.1. Residual K-Means

In this method, we first initialize the means of a regular HMM with M Gaussian components per state, and we run five iterations of K-Means. Then, we go over the training data, and for each feature vector, we find the Gaussian that is closest to it, in the Euclidean sense. We subtract the mean of the Gaussian from the feature vector, and we use the residual frames computed in this way to initialize a regular HMM with N Gaussians per state. We follow by running five passes of K-Means, using always the residual frames. The means from the first sequence of K-Means are the initial estimates for the μ_{si} parameters, and the means from the second sequence of K-Means initialize the impulse means m_{sj}.

3.2. Deconvolution from MN-bin HMM

Another way to get initial estimates for the parameters of the convolutional model, is to initialize and run K-Means for a regular HMM with MN Gaussian components per state, and then deconvolve the resulting model, using Eq. 15. In other words, we solve for the M-component Gaussian mixture and N-component impulse mixture whose convolution is the closest to the original MN-component Gaussian mixture, for each state. This method brings up the problem of the optimal assignment of Gaussians. It is clear that if we change the order of the MN Gaussians within a mixture, then Eq. 15 will result in different solution for the parameters μ_{si} and m_{sj}. Thus, it is important that we explore as many permutations as possible, before initializing the convolutional model parameters with Eq. 15. Since the number of permutations to explore is proportional to $(MN)!$, we cannot use brute force. Instead, we chose to search for the optimal assignment of Gaussians using a Genetic Algorithm (GA).

3.3. GA search for optimal deconvolution

As described in [2] and [3], the key elements of a GA are the representation of each member in the genetic population, the fitness function that assigns fitness values to each member, and the genetic operators that combine existing members to generate new ones.

In our implementation, each member is represented as a sequence of pairs (i, j), where $i, j \in [1, MN]$. Each pair (i, j) indicates a swapping of the Gaussians in the ith and jth bins. Thus, a sequence of pairs represents a permutation of the MN Gaussians within a mixture. To obtain a fitness value for each member, we order the Gaussians according to the representation of the member, and we solve for the means μ_{si} and m_{sj}, using Eq. 15. Then, we compute the Euclidean distance between the ordered MN means, and the convolved means, weighted by the number of frames assigned to each Gaussian.

The genetic operators that we use are selection, single-point crossover, and mutation. Selection is done probabilistically, based on the fitness values of the members. Crossover and mutation are applied with probabilities that are given as parameters to the search algorithm. The members of the first population (generation 0) are initialized randomly. Then, successive generations are created as a result of the genetic operators. The search is terminated when the majority of the population has converged to a local optimum.

4. COMPUTATIONAL REQUIREMENTS

It is clear from section 2 that the storage requirements for the parameters of an $M \times N$ convolutional model are $\mathcal{O}(M + N)$, since for each state we need to store the M Gaussian means and variances, and the N impulse means. However, in order to compute the likelihood in both training and recognition, we need to evaluate MN Gaussians per state, so the time complexity for likelihood computation is $\mathcal{O}(MN)$.

The time complexity for recognition can be significantly reduced, if we take advantage of the structure in the $M \times N$ convolutional model. We have found that in the case where N is relatively small compared to M, the means of the N impulses are concentrated around zero, with small variance. This suggests the following approximation for computing the observation probability of frame x in state s:

- evaluate the M Gaussians in state s, at frame x ($\mathcal{O}(M)$ time)
- find the best k Gaussians, where k is small compared to M
- convolve the best k Gaussians with the N impulses, resulting in kN Gaussians
- evaluate the kN Gaussians at frame x and compute the final probability ($\mathcal{O}(kN)$ time)

This approximation reduces the Gaussian computation to $\mathcal{O}(M + kN)$. In the next section we show that this speedup comes with essentialy no degradation in recognition accuracy.

5. EXPERIMENTAL RESULTS

In this section we present the results of recognition experiments that we conducted in order to evaluate the efficacy of the convolutional models, using the BYBLOS transcription system [4]. For the training of our models, we used approximately 24 hours of speech from the 1997 Hub-4 Broadcast News corpus. For testing, we used the 1996 Hub-4 Unpartitioned Evaluation Development test set (H4D96-UE). All models are trained based on our gender-dependent, triphone State Clustered Tied Mixture (SCTM) non-crossword system.

Table 1 shows the effect of the K-Means initialization method on the accuracy of the convolutional model, when $M = 32$ and $N = 2$.

We can see that without searching for the optimal assignment of Gaussians, deconvolution from a 64-bin model is much worse than the residual K-Means approach. Using the GA to find a better deconvolution results in accuracy that is slightly better than the residual initialization.

Initialization Method	Word Error %
Residual K-Means	33.9
32 $\times$ 2 Deconvolution (random assignment)	35.0
32 $\times$ 2 Deconvolution (GA search)	33.8

Table 1: Effect of K-Means initialization on the accuracy of convolutional models. Results are shown for the male speakers in H4D96-UE

In Table 2 we show how the convolutional model with $M = 32$ and $N = 2, \ldots, 4$ compares to the standard 32-bin and 64-bin models, in terms of accuracy and speed. All the convolutional models in this table are initialized using the residual K-Means method.

Model Type	Word Error %	xRT in bw pass
Standard 32-bin	32.8	12
Standard 64-bin	32.3	22
32 $\times$ 2	32.1	23
32 $\times$ 3	32.1	34
32 $\times$ 4	31.9	45
32 $\times$ 4 fast	32.0	24

Table 2: Effect of varying the number of impulses on the accuracy and speed of convolutional models.

First, we can see that there is a 0.5% absolute gain for the 64-bin model, compared to the 32-bin baseline. This is expected, since the 64-bin model has twice as many parameters as the 32-bin model. With a 32×2 model, however, we can obtain a 0.7% gain over the 32-bin baseline, and 0.2% absolute over the 64-bin model, by just increasing the number of parameters to 34 per state (32 Gaussian means/variances, and 2 impulse means). Further increase of the number of impulses to 4 gives us an additional 0.2% gain. The last line in Table 2 shows the performance of the 32×4 model when we use the approximate Gaussian computation described in section 4, with $k = 5$. We see that there is approximately a factor of two speedup in decoding time, with essentially no loss in accuracy.

6. CONCLUSIONS AND FUTURE RESEARCH

We have presented a novel approach in modeling the residual frame variability in speech, using convolutional densities. We described in detail the case where the density is the convolution of an M-component Gaussian mixture with an N-component impulse mixture, and showed two methods for initializing the parameters of the convolutional model, the residual K-Means, and the deconvolution from a MN-bin model. In the latter case, we showed that it is important to use a search method for finding a good permutation of the MN Gaussians in each state, before applying the deconvolution. Using a GA for search of the optimal permutation results in accuracy comparable to the residual K-Means approach. We described a method for reducing the computational requirements of the convolutional model during recognition, from $\mathcal{O}(MN)$ to $\mathcal{O}(M + kN)$, where $k < M$, without sacrificing accuracy. Finally, we showed results on the H4D96-UE demonstrating that the 32×2 and 32×4 models achieve better performance than the standard 64-bin model, with fewer parameters.

We are currently working on improving the accuracy of the convolutional models, by trying different ways of initialization. In particular, we are interested in the use of Parametric Trajectory Models [5], [6] as a means of dividing the data into N clusters; we then plan to initialize an M-component Gaussian mixture for each cluster, and combine the mixtures into a $M \times N$ convolutional model.

Another area of research is the adaptation of the convolutional models. Both Speaker Adaptive Training (SAT) and unsupervised MLLR adaptation on the test are expected to give additional gain for the 32×2 model compared to the 64-bin baseline, due to the smaller number of parameters in the convolutional model.

7. ACKNOWLEDGMENTS

This work was supported by the Defense Advanced Research Projects Agency and monitored by Ft. Huachuca under contract No. DABT63-94-C-0063 and by the Defense Advanced Research Projects Agency and monitored by NRaD under contract No. N66001-97-D-8501. The views and findings contained in this material are those of the authors and do not necessarily reflect the position or policy of the Government and no official endorsement should be inferred.

8. REFERENCES

[1] Rabiner, L. "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of the IEEE*, Vol. 2, pp. 257-285, February 1989

[2] Mitchell, Tom M. "Machine Learning", pp. 249-270, McGraw-Hill, 1997.

[3] Goldberg, D. "Genetic algorithms in search, optimization, and machine learning", Addison-Wesley, 1989.

[4] Kubala, F. et al, "The 1997 BBN Byblos Hub-4 Transcription System", *Proceedings of the Broadcast News Transcription and Understanding Workshop*, February 8-11, 1998, Lansdowne, VA.

[5] Gish, H. and Ng, K., "A Segmental Speech Model with Applications to Word Spotting", *Proceedings of the ICASSP*, Vol. 2, pp. 447-450, 1993.

[6] Gish, H. and Ng, K., "Parametric Trajectory Models for Speech Recognition", *Proceedings of the ICSLP*, Vol. 2, pp. 466-469, 1996.