SEQUENTIAL CLUSTERING ALGORITHM FOR GAUSSIAN MIXTURE INITIALIZATION

Ronaldo MESSINA and Denis JOUVET

France Télécom R&D – DIH/IPS
2, Av. Pierre Marzin
Lannion 22307, France

ABSTRACT

A simple sequential algorithm for deriving initial values for Gaussian mixture parameters used in HMM-based speech recognition is presented. The proposed algorithm sequentially clusters the training frames, in the order in which they are available and according to the density to which they are associated. This frame-density association results from a frame-state alignment of the training data performed with a single-Gaussian model, which is good enough for such a force-alignment task. The models obtained with the proposed sequential clustering procedure provide good speech recognition performance when compared to models obtained with the usual Gaussian splitting procedure.

1. INTRODUCTION

Mixture of Gaussians (MG) are used in Hidden Markov Models (HMM) as an approximation to the state-dependent observation densities. The use of MG provides fine acoustic resolution that is necessary for high recognition performance, but there is no known solution to decide how many elements a mixture should have. The number of elements also depends on the kind of data that is modeled: landline/wireless telephone speech can be very different from each other, female/male vowel-like sounds, mixed SNR values, etc.

There are two main approaches to determine the number of elements in the mixture; we assume that single Gaussian models (SGM) are available:

- each Gaussian, or the one with maximum variance, is split in two [1] by a random perturbation of the mean vector and then a few training iterations are performed to update the parameters; this process is repeated until a termination criterion is satisfied; and
- a subset of the training data is aligned with the SGM and a k-means algorithm is used to cluster the data of each state into a target number of clusters; each cluster represents one element in a mixture.

There are also several criteria available [2] to stop growing the mixture:

- when the number of frames used to estimate the Gaussian element parameters falls below a given threshold;
- when the likelihood increase falls below a pre-set minimum; and
- the Bayesian Information Criterion (BIC) [3]; the BIC controls model complexity by penalizing the likelihood with the number of parameters; if the BIC gain is negative or below a minimum threshold, the model with the smallest number of parameters is retained.

This paper proposes a simple algorithm that goes sequentially through the training data, retrieving the aligned state for each frame; if a distance criterion (c.f. section 4) is not satisfied the corresponding mixture is increased, otherwise the nearest Gaussian component is updated (sequential clustering). The state alignment is obtained via forced-Viterbi decoding with SGM which are relatively easy to initialize and train. The same alignment is also used to train the resulting MG models; this training procedure is the well known segmental k-means training [4] (SKM), but the Viterbi-alignment is performed only once and not at each iteration. Avoiding alignment of training data at each iteration greatly reduces the total training time.

The rest of the paper goes as follows. The speech recognition system used in the experiments and the training and test corpora are briefly described. Then the sequential clustering algorithm is exposed in detail. Next the experiments and recognition results are presented and, finally, we conclude with a discussion of the proposed algorithm.

2. SYSTEM DESCRIPTION

Our speech recognition system uses continuous density HMM (a density is either a single Gaussian or a mixture of Gaussians) and the modeling units are context-dependent phones. Sharing of the acoustic parameters between the different context-dependent units associated to the same phone limits the total number of parameters [5]. Only diagonal covariance Gaussian functions are used. The features are the frame energy, the first nine MFCC [6] and their first and second-order derivatives [7]; blind equalization is used to reduce channel influence on the features.

We first train context-independent models (CI) which are used to bootstrap context-dependent models (CD). The topology of the CI-HMM is quite simple, as shown in figure 1. CD models use the same topology, but there are as many entry/exit states and densities as different left/right contexts. There are 36 CD phoneme models (French) and 3 CI models for silence and short pauses.

3. SPEECH CORPORA

3.1. Training

for training models we use a collection of in-house corpora that amounts to about 300h of speech data. The utterances are a mix of isolated words and short phrases and cover most of the contextual variability in the French language.

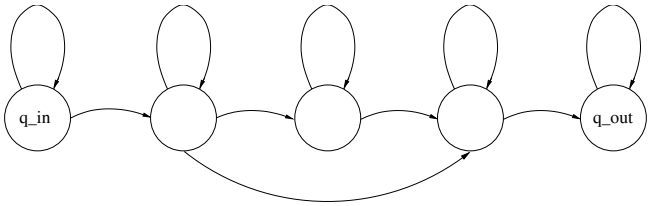

Fig. 1. CI models topology.

3.2. Testing

We evaluate the models in some common tasks:

- *Commands*: isolated words or short phrases, such as "Perros-Guirec" or "Autre rubrique" (12778 utterances);
- *Digits*: isolated digits (4735 utterances);
- *Numbers*: numbers (NN), from 00 to 99 modeled as one word, e.g. "10: *dix*" or "99: *quatre vingt dix-neuf*"(7244 utterances); and
- *TelNumbers*: telephone numbers, in the usual french forms: 0 80X NN NN NN or NN NN NN NN NN (5664 utterances), where X is a digit from 0 to 3.

For *Numbers* and *TelNumbers*, each pair (NN) counts as a single compund word for error counting; thus recognizing 66 *soixante six* in the place of 70 *soixante dix* counts as one error.

4. SEQUENTIAL CLUSTERING ALGORITHM

We assume that the training data is already aligned to some models [1]. It is sufficient to align the data to single Gaussian models; single Gaussian models do not deliver good performance in recognition but with forced-alignment they provide good enough segmentation of the data which allows to derive proper initial models as will be shown by the results.

The idea driving the algorithm is to increase a mixture (i.e. add a Gaussian element) when the distance between a frame and the mixture is above a given threshold. We can set threshold values for the maximum number of elements in a mixture and the appartenance distance to control the size of the mixtures. Proceeding in this way avoids splitting the Gaussian elements and then performing some training iterations to updated the parameters and also avoids reading large amounts of data in memory. Each mixture-density holds for each possible Gaussian element: a mean vector, an accumulator (for the variances), and the number of frames k that were used to estimate the parameters. When a new element is added, we set the mean vector to be equal to the current frame and the variance accumulator is set to zero. For each mixture it is given the maximum number of elements $maxElem$ and the maximum distance $dMax$ to accept the frame as belonging to the mixture. The distance between a frame $\mathbf{x}$ and a mixture, is given by the minimum euclidean distance between the frame and the mean vectors; let $d_j^t = \mathrm{dist}(\mathbf{x}, \mu_j^t)$ denote the distance to element j in the t-th mixture, and let i denote the nearest element and $\mathrm{numElem}(t)$ the actual number of elements in the mixture. We do not use the variances because in the beginning there are no variances and it is not always garanteed that each element has seen enough frames to allow confidence in the estimated variances. It is possible to set a second distance threshold and use the Mahalanobis distance when all elements have seen more than a minimum number of frames; informal tests showed that there was no gain in using Mahalanobis distance and more, this would introduce yet another threshold to tune. To avoid numerical problems, the mean and the variance's accumulator are calculated as described in [8]. After reading k frames (current k-th frame is $\mathbf{x}$) for a given mixture element i the mean for dimension d is updated:

$$\mu_{i\,d}^{\,k} = \mu_{i\,d}^{\,k-1} + \frac{x_d - \mu_{i\,d}^{\,k-1}}{k} \tag{1}$$

And the value of the variance accumulator:

$$S_{i\,d}^{\,k} = S_{i\,d}^{\,k-1} + \left(x_d - \mu_{i\,d}^{\,k}\right) \cdot \left(x_d - \mu_{i\,d}^{\,k-1}\right) \tag{2}$$

The variance is given by $\sigma_{i\,d}^{2\,k} = \frac{S_{i\,d}^{\,k}}{k-1}$. The algorithm is described below.

---

for all pairs frame/density: $(\mathbf{x}; t)$ **do**
  $i = \arg\min_j d_j^t$
  $d_i^t = \mathrm{dist}(\mathbf{x}, \mu_i^t)$
  if $d_i^t < dMax$ **then**
    update element i
  else
    if $\mathrm{numElem}(t) < maxElem$ **then**
      increase mixture
    else
      update element i
    end if
  end if
end for

After processing all training data, the Gaussian elements that saw less than a minimum number of frames $minM$ could be removed, or the nearest element can be found and these elements are merged. We chose the latter approach, and we keep merging as long as there is still an element with small mass. For densities that are associated to less frames than $minM$ we set the maximum number of elements to one beforehand; we do this to avoid growing a mixture and then merging the elements back into a single element.

This provides the initial values of Gaussian parameters and the mixture weight of each Gaussian is set to the element mass divided by the sum of the elements' masses. The resulting models can then be trained with SKM or Estimation-Maximization iterations.

4.1. Modifications

The cost of distance calculation could (possibly) be reduced by calculating the distance to the first element in the mixture and use this value as a threshold to abort the distance calculation for the remaining dimensions. As the distance always increases with the dimension, we can check its value at every few dimensions and abort the calculation as soon as possible.

The influence of outlier frames (if detected by the forced alignment) can be controlled with the $minM$ value or by changing the update policy. It is expected that the elements representing outlier frames will have smaller mass than the other element, thus setting

[1] The target models' topology could be different, but the labels identifying the densities have to be transformed accordingly.

$minM$ to a high value will remove those elements; or we can decide to give more importance to outliers by not updating the mean value for points that are within a ball with radius r around the mean. We evaluated the approach that favorizes the outliers but there was no significant difference in the resulting models.

5. EXPERIMENTS

5.1. Training data alignment

First we train CI models with standard SKM (aligning at each iteration), with a subset of the training data (a phonetically rich subset, with about 13h of speech) and use these models to bootstrap CD models that are then trained (standard SKM iterations) using the full database. The training data is then aligned to the CD-SGM and this aligment is kept for the remaining of the training process.

5.2. Baseline models training

The Gaussians of the SGM are split (only if the mass is greater than 50) and then a few more SKM training iterations are performed with the fixed alignment. This split/train procedure is repeated to obtain models with 8 and 16 Gaussians per density, that will be used as baseline for comparisons (2 and 4 Gaussian models are less interesting). Relying on the alignment obtained with simple models may look dangerous as we can never recover from a "bad" alignment. The alignment can always be "refreshed" with the (better) trained models and the initialization/training process can be performed again to estimate (better) initial models. It is also possible to refresh the alignment and continue the training without performing the initialization process again. We included the results with the SGM used to align for illustration purposes.

Table 1 shows the total number of Gaussians for the baseline models and the identifier used to present the recognition results.

expID	max #Gauss/dens.	#Gauss	avg # Gauss/dens.
base01g	1	3180	1
base08g	8	20274	6.4
base16g	16	37543	11.8

Table 1. Model sizes for different number of Gaussians/density.

The maximum number of Gaussian elements per density is not attained due to the limited amount of training data. Ideally, increasing the amount of training data would allow the densities to have the desired maximum number of elements.

5.3. Sequential clustering mixture initialization

We ran the sequential clustering algorithm to obtain initial models with a maximum of 8, 16 and 64 Gaussians per density. In table 2 we present the values of the thresholds and the total number of Gaussians in the models with the identifier that is used when presenting the results. The line *g64_red* will be explained later.

Threshold ($maxElem$, $dMax$ and $minM$) values were chosen to have a total number of Gaussians not too different than from those in the baseline models and also to sample the resulting model sizes.

expID	$maxElem$	$minM$	$dMax$	#Gauss
g8m20d5	8		5	18826
g8m20d10			10	18764
g8m20d20			20	18008
g8m20d33			33	15128
g8m30d17		30	17	17885
g8m50d17		50	17	17255
g16m20d5	16		5	34225
g16m20d10			10	34067
g16m20d20			20	31566
g16m20d33			33	23900
g16m30d17		30	17	31517
g16m50d17		50	17	30112
g64m20d5	64	20	5	107363
g64_red	64	—	—	37543

Table 2. Threshold values and total number of Gaussians.

6. RESULTS

In table 3 the baseline recognition results (word error rate – WER) for the described tasks are presented. The results obtained with the initial models resulting from the presented algorithm are shown in table 4 and after training (5 SKM iterations) in table 5.

expID	Task			
	Commands	*Digits*	*Numbers*	*TelNumbers*
base01g	1.72	2.60	8.23	15.29
base08g	1.12	1.46	5.14	11.58
base16g	1.05	1.10	5.00	10.85

Table 3. Recognition results (%WER) for the baseline models.

The results show that the initial models have worse performance in the "hard": tasks *Numbers* and *TelNumbers* when $minM$ increases, but after training the performance is comparable; performing a few extra training iterations brought the results to the confidence interval (results not presented). The other tasks are easier and there is smaller gain with further training. This result is expected, because having less Gaussians in total reduces the acoustic resolution of the initial models. The value of $dMax$ has stronger influence on the final number of Gaussians and setting it too high will cause the models to be "blurred", as more frames are absorbed by each element instead of increasing the number of elements in the mixture. In the case of larger mixtures (16 Gaussians/density), reducing $dMax$ increases performance comparing to 8 Gaussian/density models; having more freedom to add Gaussians, could possibly tune the extra Gaussians to particular details that increase performance. The models with 64 Gaussian/density have an enormous number of Gaussians; we later reduced this number by finding the pairs of Gaussians that leads to the smallest drop in likelihood when merging those two Gaussians into one. We merged the Gaussians down to the same total number as for the baseline 16 Gaussians/density models (table 2, last row). The results for the reduced Gaussian pool models are presented in table 6, for models just after merging and after a few SKM iterations.

The results show that we can start from larger densities (the cost is moderate) and then reduce the Gaussian pool to a more

expID	Task			
	Commands	*Digits*	*Numbers*	*TelNumbers*
g8m20d5	1.22	1.71	5.60	13.54
g8m20d10	1.25	1.61	5.54	13.85
g8m20d20	1.24	1.61	5.84	13.14
g8m20d33	1.24	1.77	6.09	13.14
g8m30d17	1.21	1.56	5.84	13.32
g8m50d17	1.21	1.56	5.94	13.82
g16m20d5	1.13	1.35	5.30	13.08
g16m20d10	1.13	1.41	5.59	13.43
g16m20d20	1.07	1.41	5.26	12.99
g16m20d33	1.15	1.63	5.45	12.78
g16m30d17	1.17	1.31	5.43	13.27
g16m50d17	1.15	1.33	5.59	13.85
g64m20d5	0.99	1.10	5.05	13.65

Table 4. Recognition results (%WER) for the initial models.

expID	Task			
	Commands	*Digits*	*Numbers*	*TelNumbers*
g8m20d5	1.12	1.44	5.26	12.01
g8m20d10	1.17	1.25	5.16	12.25
g8m20d20	1.27	1.44	5.36	12.18
g8m20d33	1.18	1.44	5.65	12.21
g8m30d17	1.21	1.35	5.48	11.69
g8m50d17	1.22	1.35	5.49	11.74
g16m20d5	1.03	1.27	4.85	10.97
g16m20d10	1.12	1.12	5.04	11.61
g16m20d20	1.10	1.18	4.93	11.72
g16m20d33	1.08	1.29	5.09	11.27
g16m30d17	1.14	1.03	4.94	11.28
g16m50d17	1.14	1.06	4.94	11.27
g64m20d5	0.92	0.70	4.21	10.26

Table 5. Recognition results (%WER) for models after training.

reasonable size and do better than starting from a limited pool (i.e. 16 Gaussians/density) or with the usual Gaussians splitting (which implies in a longer training time).

7. CONCLUSION

We presented a simple algorithm to initialize mixtures of Gaussian densities for HMM speech recognition; initial models of good quality are obtained with one pass over the full training database. The initial models deliver reasonable performance because all training data is seen, instead of a subset as would happen with a k-means initialization. The advantage over Gaussian splitting is the shorter training time; the cost of initialization is roughly the same as one iteration with fixed alignment.

Initial models obtained with the presented algorithm already give reasonable performance considering that any training iteration was performed. After a few training iterations, the results are comparable or even better (but not at a significant level) than the baseline. Starting from densities with a larger number of elements, performing some training iterations, then reducing the Gaussian pool size by merging pairs of similar Gaussians and tuning the parameters with a few more training iterations resulted in better models than Gaussian splitting and starting from smaller densities. This is another advantage of the proposed algorithm, to rapidly produce good initial models based on a large number of Gaussian functions.

expID	Task			
	Commands	*Digits*	*Numbers*	*TelNumbers*
g64_red	1.06	1.03	4.76	10.65
g64_red_trn	1.06	0.89	4.62	10.32

Table 6. Recognition results (%WER) for reduced models.

It should be reminded that it is not necessary to model exactly the distribution of the training data, as this could result in "over-training", i.e. the models recognize only the training data. It is sufficient that the densities give a non-floored value for the probability of an observation when it belongs to the correct search hypothesis. In the future, we will investigate alternative ways to construct the mixtures in a fashion that increases the probability for the correct density with respect to the other competing densities.

8. REFERENCES

[1] Ananth Sankar, "Experiments with a Gaussian Merging-Splitting Algorithm for HMM Training for Speech Recognition," in *Proceedings of the DARPA workshop*, 1998, pp. 99–104.

[2] Y. C. Chan, M. Siu, and B. Mak, "Pruning of state-tying tree using bayesian information criterion with multiple mixtures," in *Proceedings of the International Conference on Spoken Language Processing*, Beijing, China, 2000, vol. IV, pp. 294–297.

[3] Gideon Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461 – 464, 1978.

[4] L. R. Rabiner, J. G. Wilpon, and B.-H. Juang, "A segmental k-means training procedure for connected word recognition," *AT&T Tech. Journal*, vol. 64, no. 3, pp. 21–40, May 86.

[5] Katarina Bartkova and Denis Jouvet, "Modelization of allophones in a speech recognition system," in *Proceedings of the XII International Congress on Phonetical Sciences*, Aix-en-Provence, August 19-24 1991, vol. 4, pp. 474–477.

[6] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on ASSP*, vol. 28, no. 4, pp. 357–366, 1980.

[7] Lawrence Rabiner and Biing-Hwang Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, Englewood Cliffs, NJ, 1993.

[8] Donald E. Knuth, *The Art of Computer Programming*, vol. 2: Seminumerical algorithms, Addison-Wesley, Reading, Massachusetts, 1997.