COMPARISON RESULTS FOR SEGMENTAL TRAINING ALGORITHMS FOR MIXTURE DENSITY HMMS

Mikko Kurimo

Neural Networks Research Centre
Helsinki University of Technology
P.O.Box 2200, FIN-02015 HUT, Finland
Tel. +358 9 451 3266, FAX: +358 9 451 3277, E-mail: mikko.kurimo@hut.fi

ABSTRACT

This work presents experiments on four segmental training algorithms for mixture density HMMs. The segmental versions of SOM and LVQ3 suggested by the author are compared against the conventional segmental K-means and the segmental GPD. The recognition task used as a test bench is the speaker dependent, but vocabulary independent automatic speech recognition. The output density function of each state in each model is a mixture of multivariate Gaussian densities. Neural network methods SOM and LVQ are applied to learn the parameters of the density models from the mel-cepstrum features of the training samples. The segmental training improves the segmentation and the model parameters by turns to obtain the best possible result, because the segmentation and the segment classification depend on each other. It suffices to start the training process by dividing the training samples approximatively into phoneme samples.

1. INTRODUCTION

The recognition task used as a test bench for the training algorithms is the speaker dependent, but vocabulary independent automatic speech recognition. The recognition is based on connecting the HMMs of the phonemes to decode the phoneme sequences of the spoken utterances. The HMMs parameters are trained from the data using a set of training words collected for each speaker. The output density function of each state in each model is a mixture of multivariate Gaussian densities and so these HMMs are called MDHMMs (mixture density HMMs).

The probability density in state i for a D-dimensional observation vector $\boldsymbol{O}_t$ ($t = 1, \ldots, T$ is a discrete time index) is computed as a linear combination

$$b_i(\boldsymbol{O}_t) = \sum_{m=1}^{M} c_{im} b_{im}(\boldsymbol{O}_t) \, . \tag{1}$$

The mixture weights in (1) satisfy the conditions $c_{im} \geq 0$ and $\sum_{m=1}^{M} c_{im} = 1$ for all states i [9]. In each Gaussian density $b_{im}(\boldsymbol{O}_t) \sim N(\mu_{im}, \Sigma_{im})$, $\mu_{im} \in \mathcal{R}^D$ and $\Sigma_{im} \in \mathcal{R}^{D \times D}$ are the mean vector and the covariance matrix, respectively.

The training of the context-independent phoneme models for the minimum recognition error rate is difficult, because the variation of the phonemes in different conditions and contexts is substantial and the output densities of different phonemes overlap. A structure flexible enough to automatically adapt to all the complicated density functions, will have a vast number of parameters and for proper estimation, the quality and quantity of the available training data is crucial. The size of the models and the training database demand robustness against initial parameter values in order to avoid excessively many training epochs and long training times.

The division of the training samples into phonemes and states leads to difficulties, if standard statistical methods would be used to estimate the states of the MDHMMs. Because the segmentation and the segment classification in the training samples depend on each other, the so-called segmental training methods [9] are popular. The idea is to improve the segmentation and the model parameters by turns to obtain the best possible result. The selection of the way to begin the training process can be crucial, if the amount of training epochs is limited and the number of parameters large.

The traditional segmental training algorithm called the segmental K-means (SKM) [10] attempts to maximize the likelihood of the optimal Viterbi segmentation for the training samples. In the parameter reestimation phase the output density of each state is estimated in order to maximize the likelihood of the corresponding segments of training data. The segmental GPD (Generalized Probabilistic Descent) [1] is chosen in this work as a representative of the more advanced segmental training methods that estimate the output density parameters aiming directly at the minimization of phoneme classification errors. This is possible by computing the corresponding segmentations for the closest rival phoneme sequences of the correct one as well. Then the parameters are estimated to minimize a general differentiable loss function determined to closely reflect the misclassification rate.

The problem in practice with the common training algorithms, the SKM and the segmental GPD (SGPD), is that they sometimes converge slowly to low error rates unless good initial models are available. One approach is to apply both methods successively, first SKM to get the initial models and then SGPD to optimize the error rate. The inconvenience in that approach is the long training time and that the training data set may thus need to be cycled quite many times before error rate is low enough.

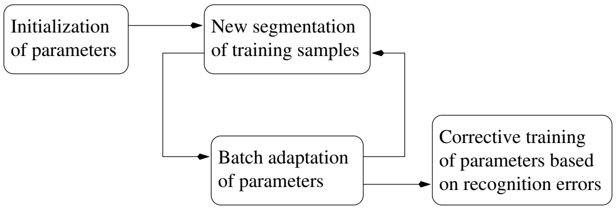

Figure 1: The phases of the HMM training.

In this work the power of the segmental training is enhanced by applying neural network methods to the output density estimation. The SOM (Self-Organizing Map) [4] is used first to improve the learning of extensive density models for large training databases in the beginning of the HMM training. The LVQ (Learning Vector Quantization) [4] is applied after that to incorporate simple discriminative learning into the parameter adjustments to decrease the phoneme recognition error rate.

2. THE SEGMENTAL SOM TRAINING

In this work the MDHMMs are applied so that the Gaussians used for the mixture density output models are tied phoneme-wise. This means that instead of having unique Gaussians for each state or one common pool of Gaussian for every state [2], the states representing the same phoneme, i.e. states that belong to the same HMM, share the same Gaussians [5]. The codebook of Gaussians for each phoneme is initialized by training one SOM for the samples of each phoneme and using then the SOM units as the mean vectors for the Gaussian kernels. It is remarkable that the SOMs are rather easy to train because the training data is required only to be approximately divided into phoneme samples without yet concerning about the states of the HMMs.

The initialization of the widths of the Gaussian kernels and the weight vectors that associate each state to each kernel can be based on the portion of the phoneme samples that will be mapped to the corresponding SOM units. The obtained initialization can be well applied for the traditional segmental training (SKM) [5], but the SOM training can be also continued further as studied in [8]. The idea is to incorporate the use of the training neighborhoods into the MDHMM training by the method called the *segmental SOM training*.

The segmental SOM and SKM have the same segmentation phase (see Figure 2) and for the parameters of the best-matching mixture densities for each training vector the adaptation parts are similar as well. The main difference is that also the parameters of mixtures belonging to the neighborhood of the best-matching component are adapted. Thus the adding of the *neighborhood function* $h_{o,m}$ [4] to the SKM leads to the batch adjustments (2) and (3). Normally, $h_{o,m} \in [0,1]$ and $h_{o,m} \to 0$ as the distance on the SOM grid between the adjusted unit m and the best-matching unit o increases. It should be noted that because the index o is a function of the discrete time index t, also the value of $h_{o,m}$ depends on t.

In the segmental SOM a batch iteration produces new mean vectors $\boldsymbol{\mu}_{jm}$ for the Gaussian kernels $N(\mu_{jm}, \Sigma_{jm}), \forall m = 1, \ldots, M_j$ of mixture density codebook j, by computing the averages of the associated sample vectors $\boldsymbol{O}_t$:

$$\hat{\boldsymbol{\mu}}_{jm} = \frac{\sum_{t=1}^{T} \delta(q_t, j) h_{o,m} \boldsymbol{O}_t}{\sum_{t=1}^{T} \delta(q_t, j) h_{o,m}} , \qquad (2)$$

where the indicator function $\delta(q_t, j) = 1$, if q_t (the decoded state for time t) is connected to the codebook j (otherwise $\delta(q_t, j) = 0$). In general, each state could use several codebooks and the same codebooks could be connected to several states. Here, the states in the same HMM use the same codebook. The most probable correct state sequences $q = q_0, \ldots, q_T$ corresponding to the observation sequences $\boldsymbol{O}$ are determined by the current segmentation. The equation (2) can be described as the batch adaptation step of the normal SOM [3] applied to a system with several codebooks.

If individual covariance matrices are needed for each Gaussian kernel, the adaptation formula for $\hat{\boldsymbol{\Sigma}}_{jm}$ corresponding to (2) can be obtained by substituting $\boldsymbol{O}_t$ in (2) by the matrix of the deviations from the mean vector $(\boldsymbol{O}_t - \boldsymbol{\mu}_{jm})(\boldsymbol{O}_t - \boldsymbol{\mu}_{jm})^T$.

The mixture weights (1) that connect individual Gaussian kernels to the output density function of a HMM state are set in the batch iteration to reflect the contribution of the kernels for the output density function of that state:

$$\hat{c}_{im} = \frac{\sum_{t=1}^{T} \delta(q_t, i) h_{o,m}}{\sum_{t=1}^{T} [\delta(q_t, i) \sum_{m=1}^{M_i} h_{o,m}]} , \qquad (3)$$

where the indicator function $\delta(q_t, i) = 1$, here, if the state q_t equals state i (otherwise $\delta(q_t, i) = 0$).

From the different kinds of neighborhood functions $h_{o,m}$ [4], the simple bubble type is used here for simplicity. After the adaptation of $\boldsymbol{\mu}_{jm}$s and c_{im}s the size of the bubble is decreased gradually until it is empty, and then the process continues by adapting only the parameters of the best-matching mixture for each training sample.

The motivation for the neighborhood adaptation is the smoothing and ordering of the codebooks. The trade-off between the smoothing and the fitting accuracy to the training data is controlled by the width of the neighborhood. A smooth density representation is important, because despite the finiteness of the training data a high level of generalization and modeling accuracy for the independent test data must be simultaneously maintained. A wide neighborhood at the beginning provides a high level of smoothing for every codebook unit and draws them into useful regions in the input space. As the neighborhood is then shrunk during the training a closer adaptation occurs in the areas well represented by the training data. Compared to the codebooks trained without smoothing (by SKM) the accuracy provided by the best-matching Gaussian is usually worse, but that of the next (N-1)-best matches will be better, however, providing better coverage for slight variations in the test data.

The motivation to have ordered density codebooks is to enable accelerated state PDF estimation. In practice, a small set of best-matching kernels tends to dominate the density estimate in a high-dimensional Gaussian mixture. Thus the densities can be well approximated by excluding the other kernels. Since the search for the N-best matches consumes a significant part of the total computational load, the search speed-ups may have a significant effect on the total recognition speed [8]. By exploiting the similarity of the successive feature vectors and the SOM topology in the mixtures, the approximate location of the N-best candidates can be determined accelerating significantly the state PDF estimation [8]. As the radius of the applied neighborhood function decreases gradually to zero the fine structure of the topology is lost due to the folding that increases the density estimation accuracy. However, some coarse structure will still be available to maintain smoothness and search acceleration capabilities.

3. THE SEGMENTAL LVQ TRAINING

The segmental LVQ3 training [6] is in many ways similar to the SGPD improving the HMM parameters iteratively by comparing the best alternative paths through the HMM states for each training sample, updating the parameters and computing new paths again. The most important differences lay in the specifications of when and how the parameters are modified in respect to the different paths.

In the segmental LVQ3 there are two optional modes depending on whether a training token (a word) would be correctly recognized by the existing models or not. If the recognition is correct, the tuning is similar as in the conventional SKM training with no discrimination. For incorrectly recognized words, the likelihood maximization is applied to the states on the path producing the correct phoneme sequence. Where the best incorrect path differs from that, the discriminative training is applied to lower the likelihoods of the incorrect states. In GPD the amount of the tuning depends normally on the exact extent of the derivative of the whole word misclassification measure [1] as well, but in LVQ3 only the relative difference of the modifiable parameter values matter. The supposed gains of this method are the avoidance of extra learning and scaling parameters that would make the learning laws complicated and extensive likelihood damping that might decline the convergence. The direct use of the whole word or sentence misclassification measure to control the magnitude of the learning actions may sometimes be misleading, because there might be both small and large misses in the same word.

The batch formulation of the adjustment of the mean vector $\boldsymbol{\mu}_{jm}$ for each Gaussian kernel $m = 1, \cdots, M_j$ of codebook j is the weighted average of all the associated data vectors, where the representatives of incorrect states have a contribution equal to the vector of the same size, but opposite direction (note that $[\boldsymbol{O}_t + (2\boldsymbol{\mu}^{old} - \boldsymbol{O}_t)]/2 = \boldsymbol{\mu}^{old}$). A data vector $\boldsymbol{O}_t$ affects to the adjustment of $\boldsymbol{\mu}_{jm}$, if the corresponding state q_t on the decoded correct path uses codebook j and the index o of the best-matching Gaussian in that codebook equals to m, thus

$$\hat{\boldsymbol{\mu}}_{jm} = \frac{\sum_{t=1}^{T_l} \delta(q_t, j)\delta_1(m = o)\delta_1(\bar{q}_t = q_t)\boldsymbol{O}_t}{\sum_{t=1}^{T} \delta(q_t, j)\delta_1(m = o)} + \frac{\sum_{t=1}^{T_l} \delta(q_t, j)\delta_1(m = o)\delta_1(\bar{q}_t \neq q_t)(2\boldsymbol{\mu}_{jm} - \boldsymbol{O}_t)}{\sum_{t=1}^{T} \delta(q_t, j)\delta_1(m = o)}, \quad (4)$$

where $\delta(q_t, j)$ is defined as in (2) and

$$\delta_1(z) = \begin{cases} 1, & \text{if } z \text{ is true}, \\ 0, & \text{if } z \text{ is false}. \end{cases} \quad (5)$$

For the mixture weight update, the idea is to increase the likelihood of a state by weight increase, if the Gaussian matches to a correct state and give a corresponding decrease for a match to an incorrect state. To avoid any weight decreasing to zero or below, the decrease of a weight is substituted by increasing the other weights by the fraction representing to their contribution (note that $\sum_{m=1, m\neq o}^{M} c_{im} = 1 - c_{io}$).

$$\hat{c}_{im} = \frac{\sum_{t=1}^{T} \delta(q_t, i)\delta_1(m = o)\delta_1(\bar{q}_t = q_t)}{\sum_{t=1}^{T}\sum_{m=1}^{M} \delta(q_t, i)\delta_1(m = o)} + \frac{\sum_{t=1}^{T} \delta(q_t, i)\delta_1(m \neq o)\delta_1(\bar{q}_t \neq q_t)\dfrac{c_{im}}{1 - c_{io}}}{\sum_{t=1}^{T}\sum_{m=1}^{M} \delta(q_t, i)\delta_1(m = o)}, \quad (6)$$

where the indicator function $\delta(q_t, i)$ is the same as in (3) and $\delta_1(z)$ as in (5).

After the adjustments, the new parameter values are used to compute the new paths again followed by the next iteration of the parameter values. The process is iterated until the selected stopping criterion is fulfilled.

The learning in the segmental training by both SOM and LVQ is made in the batch mode, where each epoch includes the entire training data. The other possibility is to use a variable learning rate parameter to relate the modifications due to different training words. A proper definition of the learning rate would be difficult, however, because the parameter changes affect to the subsequent word segmentations.

4. EXPERIMENTS AND RESULTS

For the comparison experiments seven speakers were selected from a database consisting of four separate recordings per speaker of a set of 350 words balanced to contain the most common phoneme combinations of the normal Finnish speech. The criteria to rank the segmental training algorithms in this paper is the obtained average recognition error rate for all the speakers. The error rate is the sum of inserted, deleted and changed phonemes divided by the correct number of phonemes in the test material which is the fourth speech recording not used in the training.

The 80-dimensional acoustic feature vectors modeled by the MDHMMs are concatenations of five sets of 15-component mel-cepstra and the RMS value computed at different time intervals around the current 16 ms window moved forward in every 8 ms [7]. The MDHMM for a phoneme consists of an uni-directional chain of five states and a pool of 70 Gaussians with a fixed diagonal covariance matrix. The densities for the states are approximated by using only the three best-matching Gaussians per codebook which does not deteriorate the accuracy too much.

In Table 4 the average error rates are given for the candidate training methods. After the initialization of the models the segmental training is performed for five epochs using the whole training data. The error rate is also given for

Init.	HMM training	70 mixt.		140 mixt.	
		5 ep	10 ep	5 ep	10 ep
KM	SKM	6.2	6.1	5.6	5.6
KM	SGPD	5.8	5.6	5.5	5.4
KM	SKM+SGPD	6.2	5.4	5.6	5.0
SOM	SSOM	5.9	5.5	5.2	4.9
SOM	SSOM+SGPD	5.9	5.1	5.2	5.5
SOM	SSOM+SLVQ3	5.9	5.3	5.2	5.0
SOM	SLVQ3	5.3	5.3	4.8	4.7
SOM	SLVQ3+SGPD	5.3	4.8	4.8	4.8
-	SLVQ3	5.7	5.4	5.3	5.1

Table 1: The average test set error rates for alternative training methods after the initialization by K-means or SOM. The MDHMM training methods are the segmental K-means (SKM), GPD, SOM and LVQ3. In the two-method combinations the first method is used for the five first epochs.

models with ten epochs of training, but in most cases the five epochs is enough and the error rate does not change significantly after that except for the segmental SOM. However, if the training algorithm is changed after the five first epochs, some improvement can be still observed.

From the three first rows of Table 4, which include the conventional training methods, the best is clearly the combination of the SKM and SGPD. The next five rows, which include the segmental SOM and LVQ3, suggest that the best way is to apply the segmental LVQ3 right after the SOM initialization. The smoothing of the codebook obtained by the initialization seems to be sufficient and there is no need to try to preserve the codebook topology further, at least to achieve the optimal error rate. In [8] it was shown, however, that the codebook topology is useful for speeding up the recognition process by using faster search methods to find the best-matching Gaussians for each codebook.

It is possible to skip the separate initialization by using MDHMMs already trained for another speaker as an initial model and continue the training with the data of the current speaker. The error rate obtained by this way (the last row in Table 4) is, however, higher than that obtained by the normal SOM initialization and the segmental LVQ3 training. The savings in the training effort are not important, because the dimple SOM initialization is quite fast. Anyhow, the MDHMM of another speaker can be used to provide the approximative initial phoneme segmentation to be used as a basis of the model initialization.

The average error rates in Table 4) are computed also for larger codebooks (140 Gaussians) where the five best-matching Gaussians are used for the density computations. For the both codebook sizes used the segmental LVQ3 gives the lowest and the SKM the highest error rates. The error rates obtained by the segmental SOM and the SGPD are quite near each other for the smaller codebooks, but for the larger codebooks the SOM seems to do better. It can be observed as well that for the models trained by the segmental SOM and LVQ3 it is possible to improve the error rate significantly by additional training by the SGPD. However, this seems to concern only the smaller codebook option. For the larger codebooks the SGPD is sometimes unstable and may even increase the number of recognition errors.

5. CONCLUSION

Different segmental training methods were compared for mixture density HMMs. The best results were obtained by training the models using the segmental LVQ3. By mixing the segmental training algorithms so that the models obtained by one is fed as an initialization to another, better results can sometimes be obtained than using the individual methods for the same amount of training epochs.

The lowest average unlimited vocabulary phoneme error rate obtained for the current test material by the context independent phoneme models was 4.7 %. In order to eliminate more recognition errors for the same task it may be necessary to apply a more suitable duration control for the phonemes and post-processing for the phoneme sequences.

6. REFERENCES

1. W. Chou, B.H. Juang, and C.H. Lee, *"Segmental GPD training of HMM based speech recognizer"*, Proc. ICASSP'92, pp. 473–476, San Francisco, 1992.
2. X.D. Huang and M.A. Jack, "Semi-continuous hidden Markov models for speech signals", *Computer Speech and Language*, Vol. 3, pp. 239–252, 1989.
3. T. Kohonen, *"Things you haven't heard about the Self-Organizing Map"*, Proc. ICNN'93 (International Conference on Neural Networks), pp. 1147–1156, Piscataway, NJ, 1993. IEEE Service Center.
4. T. Kohonen, "Self-Organizing Maps". Springer, Berlin, 1995.
5. M. Kurimo, *"Hybrid training method for tied mixture density hidden Markov models using Learning Vector Quantization and Viterbi estimation"*, Proc. NNSP'94 (IEEE Workshop on Neural Networks for Signal Processing), pp. 362–371, Ermioni, Greece, 1994.
6. M. Kurimo, *"Segmental LVQ3 training for phoneme-wise tied mixture density HMMs"*, Proc. EUSIPCO'96, pp. 1599–1602, Trieste, Italy, 1996.
7. M. Kurimo, "Training mixture density HMMs with SOM and LVQ", Technical Report A43, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland, 1997.
8. M. Kurimo and P. Somervuo, *"Using the Self-Organizing Map to speed up the probability density estimation for speech recognition with mixture density HMMs"*, Proc. ICSLP'96, pp. 358–361, Philadelphia, PA, 1996.
9. L.R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition", *Proceedings of the IEEE*, Vol. 77, pp. 257–286, 1989.
10. L.R. Rabiner, J.G. Wilpon, and B.H. Juang, "A segmental K-means training procedure for connected word recognition", *AT&T Technical Journal*, Vol. 64, pp. 21–40, 1986.