USING THE SELF-ORGANIZING MAP TO SPEED UP THE PROBABILITY DENSITY ESTIMATION FOR SPEECH RECOGNITION WITH MIXTURE DENSITY HMMS

Mikko Kurimo and Panu Somervuo

Helsinki University of Technology, Neural Networks Research Centre
Rakentajanaukio 2 C, FIN-02150, Espoo, FINLAND
tel: +358 9 451 3266, fax: +358 9 451 3277, email: mikko.kurimo@hut.fi

ABSTRACT

This paper presents methods to improve the probability density estimation in hidden Markov models for phoneme recognition by exploiting the Self-Organizing Map (SOM) algorithm. The advantage of using the SOM is based on the created approximative topology between the mixture densities by training the Gaussian mean vectors used as the kernel centers by the SOM algorithm. The topology makes the neighboring mixtures to respond strongly for the same inputs and so most of the nearest mixtures used to approximate the current observation probability will be found in the topological neighborhood of the "winner" mixture. Also the knowledge about the previous winners are used to speed up the the search for the new winners. Tree-search SOMs and segmental SOM training are studied aiming at faster search and suitability for HMM training. The framework for the presented experiments includes melcepstrum features and phoneme-wise tied mixture density HMMs.

1. INTRODUCTION

Many problems in speech recognition originate from the complicated and overlapping probability densities of the acoustic features of different phonemes and their internal states. The treatment of these densities with a satisfactory accuracy requires a huge amount of parameters either in the form of modelling several output streams or long feature vectors incorporating several short-time features.

The efforts to decrease the recognizer error rate, in practice, leads to balancing with the increasing number of parameters so that the parameters can still be reliably estimated using the available training database and that the recognizer can operate at an acceptable speed. A remarkable part of the whole recognition time (clearly over 50% for the current system) is consumed in the search for the best-matching mixtures and for larger systems the portion increases even more. With large mixture density codebooks it is often a sufficient density approximation to compute only a few mixtures (K-best) closest to the current observation, because the other mixtures will produce such a low and inaccurate likelihood values that they have practically no effect on the search for the optimal path [7, 1, 2].

The partial distance computation (i.e. the distance computation is ordered approximately according to the decreasing variance of the vector components and stops when the reference distance is exceeded) used to speed up the winner search for long vectors gains substantially from an intelligent search order, i. e. starting from good winner candidates. The mixture density codebooks are here trained by the Self-Organizing Map (SOM) [4] so that the expectations to find the K-best matches are highest in the topological neighborhood of the best match. The best match is also not likely to move far away from the neighborhood of the previous best match, since the spectral feature vectors computed from consecutive speech samples at short time intervals often resemble each other. Thus, the suggested search strategy is to start from the neighborhood of the previous best match and gradually increase the search area towards the direction of the improving matches using the SOM topology. Similar SOM properties has been utilized, e. g. in speech coding [10]. To catch larger feature changes a full search is performed at regular time intervals allowing still significant time savings by increasing the interval at the expense of higher degree of the approximation. A related search technique was used to speed up the training of a large SOM in [5] when the best-matching unit for each recycled training sample was approximated by the corresponding best-match on the previous training epoch.

Another way to reduce the search area is to cluster the Gaussians after the training and using the obtained cluster centers to determine the exact search area [2]. This can also be done *during* the training by organizing the Gaussians initially into a tree structure and maintaining the topological ordering in the structure by training the models with the Tree-search SOM [6].

The phoneme models in the current system are phoneme-wise tied mixture density HMMs (MDHMMs) [8], where the states of the same phoneme use the same set (codebook) of Gaussian mixtures, but each state has an unique set mixture weights. Since the search for best matching mixture densities is performed for each phoneme codebook at each time step, it is natural that only a fraction of the codebooks will produce large likelihoods, i. e. small quantization errors, at a time. The states using distant density codebooks will get insignificant likelihoods and thus very low probability to be chosen to the optimal path. The actual values of their density functions will not then be important and they can be sufficiently approximated by, for example, using the mixtures in the neighborhood of previous best matches saving a lot of computations. One criterion to determine, which codebook could be roughly approximated and which not, is presented in this paper.

2. SOM FOR MDHMMS

2.1. Normal SOM

The SOM is vector quantization algorithm to project a continuous high dimensional multivariate probability function $p(\boldsymbol{x})$ into a finite set of vectors in a low dimensional grid maintaining the input space topology. Each sample $\boldsymbol{x}$ from $p(\boldsymbol{x})$ will be mapped to the nearest vector of the finite set (*codebook*) $\{\boldsymbol{m}_i\}_{i=1}^N$.

The codebook is trained iteratively by presenting input samples in a random order one at a time and adjusting the nearest codebook vector $\boldsymbol{m}_c$ (*best-matching unit*, bmu) and its neighbors closer to $\boldsymbol{x}$:

$$\boldsymbol{m}_i(t+1) = \boldsymbol{m}_i(t) + h_{ci}(t)[\boldsymbol{x}(t) - \boldsymbol{m}_i(t)] , \quad (1)$$

where $t = 0, 1, \cdots$ is a discrete time index. The function $h_{ci}(t)$ determines the *neighborhood* around the bmu and it's value can be, for example, the *learning rate* $\alpha(t) \in (0, 1)$, if the array distance between $\boldsymbol{m}_i$ and $\boldsymbol{m}_c$ is smaller than *neighborhood radius* $r(t)$ and 0 otherwise [4]. $\alpha(t)$ and $r(t)$ are gradually decreased as t is increased to obtain the desired convergence.

The learning rule (1) minimizes the expected *quantization error* using the Euclidean distance

$$E = \int ||\boldsymbol{x} - \boldsymbol{m}_c||^2 p(\boldsymbol{x}) d\boldsymbol{x} , \quad (2)$$

where $\boldsymbol{m}_c$ is determined by

$$c = \arg\min_i \{||\boldsymbol{x} - \boldsymbol{m}_i||\}. \quad (3)$$

2.2. Tree-search SOM

The reduction of the search space to speed up the winner search in SOM can be performed by training some smaller upper layers over the main SOM. The layers are constructed in order to facilitate the tree search, in which only a small upper layer SOM is checked completely and its winner guides to the essential area in the larger SOM, where the real bmu is expected to be found [6].

The tree structure gives the computational complexity $O(\log N)$ for the winner search from N SOM units. In practise, the number of distance computations is $N_0 + (L-1) * S$, where N_0 is the size of the top layer, L the number of layers and S the size of the search area on the lower SOM layers. Due to the smoothness of the SOM mapping a large S improves the accuracy of the winner search by the expense of more distance computations.

The training of the Tree-search SOM occurs quite like in the normal SOM, one layer at a time starting from the topmost smallest layer. The main difference is that the upper layer, which has then already been trained and fixed, determines the constant size neighborhood to be trained (child nodes, see Figure 1) for the lower layer.

The K-best search necessary for HMMs using SOM trained mixture density codebooks can be applied in the same manner as the winner search with Tree-search SOM. If the lowest layer search area is too small to provide all the K mixtures, several rival best-matches from

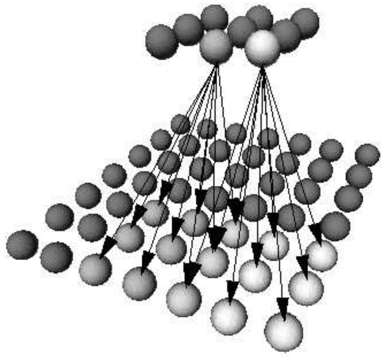

Figure 1: Two layers of the Tree-search SOM. In this map, each upper layer unit has a grid of 9 child units.

the upper layer and their associated search ares can be checked. Another possibility is to gradually increase the neighborhood on the lower layer towards the direction of the increasing responses, but the former method worked better here, however.

2.3. Density approximation by SOM

Because the SOM distributes the codebook vectors in order to approximate the input density $p(\boldsymbol{x})$ by minimizing (2), the expected distance between an observation vector $\boldsymbol{x}$ and the best-matching SOM unit is inversely proportional to the approximative density $\hat{p}(\boldsymbol{x})$. The using of the SOM for density estimation can be formulated as a Parzen estimator, where each SOM unit $\boldsymbol{m}_i$ forms a Parzen kernel with a Gaussian kernel function γ_i. To make the density approximation continuous and increase the accuracy, the weighted average of the estimates given by the different kernels can be used as the final density estimate

$$\hat{p}(\boldsymbol{x}) = \sum_{i=1}^{N} c_i \gamma_i(\boldsymbol{x}) , \quad (4)$$

where the sum of the weights c_i is one.

The obtained self-organizing reduced kernel density estimate [3] matches the mixture Gaussian density model of MDHMMs, if the mean vectors $\boldsymbol{\mu}_i$ and covariance matrices $\boldsymbol{\Sigma}_i$ of the mixtures [11] are used as the kernel centers $\boldsymbol{m}_i$ and smoothing parameters h_i [3] of the Gaussian kernel functions γ_i, respectively .

2.4. Segmental SOM training

The topology for the mixture density codebooks is obtained by initializing the mean vectors of the Gaussian mixtures by the reference vectors from SOM codebooks [8]. The HMM parameter training by segmental K-means [11] or segmental LVQ3 [9] destroys partly the smoothness of the hyperplane of the units, but leaves some rough structure, which preserves the topology in a large scale allowing the use of the probability approximations in some extent.

However, it is possible to apply the SOM training ideas to the training of HMMs by intergrating one Batch SOM epoch to each segmentation-estimation phase of the Viterbi training. The practical difference to the conventional Viterbi training is the extension of the density updates to concern also to the parameters of the topological neighbors of each best-matching mixture. This is hoped to maintain the finer SOM structure longer in the training of the mixture density HMMs. Anyhow, the neighborhood radius $r(t)$ must be decreased to zero for the last batch epochs to allow more flexibility and accuracy to the phoneme modelling.

For the mixture density HMMs the mixture mean vectors $\boldsymbol{\mu}_{im}$ and corresponding mixture weights c_{im} for each state i and its mixtures $m = 1, \cdots, M_i$ are updated by the observation sequences $\boldsymbol{O}^l$ corresponding to each training word using the segmentation (state path q^l) of the current Batch SOM epoch, are

$$\hat{\boldsymbol{\mu}}_{im} = \frac{\sum_{l=1}^{L}\sum_{t=1}^{T_l} \eta_t^l(i,m)\boldsymbol{O}_t^l}{\sum_{l=1}^{L}\sum_{t=1}^{T_l} \eta_t^l(i,m)} \quad (5)$$

$$\hat{c}_{im} = \frac{\sum_{l=1}^{L}\sum_{t=1}^{T_l} \eta_t^l(i,m)}{\sum_{l=1}^{L}\sum_{t=1}^{T_l}\sum_{m=1}^{M_i} \eta_t^l(i,m)}, \quad (6)$$

where L is the number of training words, T_l the length of the current word in frames and indicator function $\eta_t^l(i,m) = 1$ only, if $q_t^l = i$ and $h_{cim} > 0$ (otherwise zero).

For the Tree-search SOMs the segmental SOM training naturally concerns the lowest layer units, while the upper layers make the search area reduction and the neighborhood separation similarly as in the normal (fixed data labeling) Tree-search SOM training.

To improve the discrimination between the phoneme models fine tuning of the models can be performed by the corrective tuning based on LVQ2 [7] or the segmental LVQ3 training [9].

3. RECOGNITION EXPERIMENTS

The speech material used for the recognition experiments consists of four repetitions of 350 Finnish words uttered by each of the seven male speakers. Speaker-dependent phoneme models are trained using the three first word sets and the last set is left for testing. The resulting phoneme error rate is the sum of changed, deleted and inserted phonemes divided by the correct number of phonemes and averaged over the seven speakers. The material is taken from the same database as in [9], but more speakers are used here to better test the robustness of the proposed density approximation methods.

The feature vectors are 20 short-time mel-cepstrum coefficients computed in every 8 ms from a 256 point FFTs of partially overlapping Hamming windows concatenated with the energy of the signal. For better accuracy these simple features could easily be connected to longer vectors using averaging, concatenation and delta features, when the presented acceleration methods would be even more crucial, but those experiments are not included in this paper.

In the experiments the mixture density codebooks for each phoneme are formed from the basis of a SOM trained using the initially segmented training samples of the corresponding phoneme [8]. The obtained MDHMM is then trained 5 batch epochs with Viterbi training by the segmental SOM and then additional 5 epochs with zero neighborhood. Although the substitution of the traditional segmental K-means or the more efficient segmental LVQ3 [9] training by the segmental SOM does not directly improve the recognition accuracy, the segmental SOM is used to better maintain the SOM topology to be able to better investigate the gains from the acceleration of the probability density estimation.

K-best approximations				
	Train $K=5$	Train $K=10$		
K	rate %	rate %	time	distances 10^3
5	7.6	7.3	640	250
10	7.5	7.3	790	270
15	7.5	7.2	920	280

		Unordered search		Ordered search	
K	rate %	time	distances 10^3	time	distances 10^3
5	7.3	760	300	640	250
10	7.3	960	300	790	270
15	7.2	1170	310	920	280

Search radius and interval approximations					
Radius	Interval	K	rate %	time	distances 10^3
5	10	5	9.7	480	130
5	8	5	9.3	490	140
5	6	5	8.7	510	160
5	10	10	9.3	600	140
5	8	10	9.0	610	150
5	6	10	8.5	630	160
7	10	5	7.7	580	200
7	8	5	7.7	580	200
7	6	5	7.7	590	200
7	10	10	7.7	700	200
7	8	10	7.7	700	200
7	6	10	7.6	710	210

Rough approximation margins				
Margin	K	rate %	time	distances 10^3
1	5	8.7	510	160
0.01	5	9.3	490	150
1	10	8.5	630	160
0.01	10	8.9	610	150

Table 1: SOM test results. The average recognition error rate as defined in the text measures the recognition accuracy. The average recognition time without the preprocessing per word in ms and the average number of thousand distance calculations per word are used to measure the recognition speed. The "Radius" restricts the search around the previously best mixture and full search is performed only for frames separated by the "Interval". The lower the "Margin", the more codebooks will be approximated roughly by the previously K-best mixtures.

The baseline model for the presented acceleration vs. accuracy comparisons is a MDHMM of 140-mixtures per phoneme, which is double the size of the current online demonstration system [9]. The goal of the experiments was to figure out, how much the different ideas and variables affect to the recognition time and error rate, in practise. The compared recognition time is the average execution time per word in milliseconds, and to see the effect in the actual number of required distance computations, the average number of thousand distance function calls per word is included. The corresponding error rate and recognition time values for the 70-mixture system are 8.1% and 360 ms/word.

The first test in Table 1 is to vary the K in the K-best mixture density approximation. Then the unordered vs. ordered search methods (used in all other tests) are compared. Next test is for two good K

values, two suitable radii for the limited search area and three different intervals to perform the full search. The last test in Table 1 is for the rough codebook approximation, when the search area radius is 5 and the full search interval 6. If the relative distance marginal is 1, it means that no risks are taken, i.e. after testing the first mixture $\boldsymbol{\mu}_j$, the current codebook is classified to the category of distant codebooks only, if $||\boldsymbol{x} - \boldsymbol{\mu}_j|| - w_j > d_r$, where d_r is a reference distance from the best-matching phoneme codebook so far and $w_j = \max_i ||\boldsymbol{\mu}_j - \boldsymbol{\mu}_i||$ the codebook width around $\boldsymbol{\mu}_j$. Marginal 0.01 gives a wider condition, where d_r is substituted by $0.01 * d_r$.

Some Tree-search SOM based MDHMM configurations were also experimented (see Table 2).

Codebook	K_t	K_r	rate %	time	distances 10^3
36 + 121	5	5	9.1	420	100
36 + 121	5	10	8.3	610	130
36 + 121	5	15	8.1	810	140
64 + 225	15	15	8.5	1050	210

Table 2: Tree-search SOM test results. "36+121" codebook means that the upper layer of the phoneme codebook has 36 and the lower layer 121 mixtures. The mixture weights were trained only for the lower layer. K_t and K_r are the K values used in training and recognition, respectively.

4. CONCLUSIONS

The ideas presented in this paper to speed up the probability density estimation for the speech recognition with MDHMMs, are based on the ordering of the mixture densities to reduce the search area and to be able to apply an intelligent search order to find faster the best-matching mixtures.

From the tested ideas and variables, the K value decrease in the K-best mixture approximation from 15 to 5 drops remarkably the recognition time, but not much the recognition accuracy. When the search for the current K-best mixtures of each codebook is ordered to start from the K-best matches from the previous time step, the acceleration of the search time, when using the partial distance computation, is clear. The widening of the full search interval, seems to drop the accuracy quite rapidly, if the search area is small, but the increase of recognition speed is insignificant. Naturally, if the search area is already large, the full search interval does not affect much. Both the recognition speed and the error rate seems to be quite subtle for the reduction of search area, which can then be used to determine the suitable compromise of speed and quality along with the K value. The application of the rough approximation for distant codebooks seems to increase the error rate too much compared to the search acceleration to be a useful method.

The Tree-search SOM based MDHMMs gives some speed improvement, but when considering the resulting increase of the recognition errors, the training of the models still requires some further studying, especially to gain from the full power of multiple layers and large codebooks. Also the decrease in the amount of the distance calculations by the tree search does not directly drop so much the total amount of computations, because in partial distance computation applied here, the cost of a distance computation depends also on the quality of the reference distance, which can be poor for an insufficiently ordered codebook.

The average recognition error used to compare the experimental results is computed without any additional error correcting methods or grammatical rules. The average recognition time per word is measured in Silicon Graphics Power Challenge server without using the parallel processing, but because the feature vector computations are excluded, the times are intended only for the comparison between the results presented here. The chosen test material is also slightly different than in [7, 8, 9] so the results are not directly comparable.

5. REFERENCES

1. Jerome Bellegarda and David Nahamoo. Tied mixture continuous parameter modeling for speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(12):2033–2045, 1990.
2. Enrico Bocchieri. Vector quantization for the efficient computation of continuous density likelihoods. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 692–695, 1993.
3. Ari Hämäläinen. *Self-Organizing Map and Reduced Kernel Density Estimation*. PhD thesis, University of Jyväskylä, Jyväskylä, Finland, 1995.
4. Teuvo Kohonen. *Self-Organizing Maps*. Springer, Berlin, 1995.
5. Teuvo Kohonen. The speedy SOM. Technical Report A33, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland, 1996.
6. P. Koikkalainen and E. Oja. Self-organizing hierarchical feature maps. In *Proceedings of the International Joint Conference on Neural networks (IJCNN)*, volume II, pages 279–285, Piscataway, NJ, 1990. IEEE Service Center.
7. Mikko Kurimo. Corrective tuning by applying LVQ for continuous density and semi-continuous Markov models. In *Proceedings of International Symposium on Speech, Image Processing and Neural Networks*, volume 2, pages 718–721, Hong Kong, April 1994.
8. Mikko Kurimo. Hybrid training method for tied mixture density hidden Markov models using Learning Vector Quantization and Viterbi estimation. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 362–371, Ermioni, Greece, September 1994.
9. Mikko Kurimo. Segmental LVQ3 training for phoneme-wise tied mixture density HMMs. In *European Signal Processing Conference*, Trieste, Italy, September 1996. (accepted for publication).
10. Eduardo Lopez-Gonzalo and Luis A. Hernandez-Gomez. Fast vector quantization using neural maps for CELP at 2400 bps. In *Proceedings of 3rd European Conference on Speech Communication and Technology*, volume 1, pages 55–58, Berlin, Germany, September 1993.
11. Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.