SOM based density function approximation for mixture density HMMs

Mikko Kurimo

Helsinki University of Technology
Neural Networks Research Centre
P.O.Box 2200, FIN-02015 HUT, Finland

Abstract

This paper explains how some properties of the Self-Organizing Maps (SOMs) can be exploited in the density models used in continuous density hidden Markov models (HMMs). The three main ideas are the suitable initialization of the centroids for the Gaussian mixtures, the smoothing of the HMM parameters and the use of topology for fast density approximations. The methods are tested here in the automatic speech recognition framework, where the task is to decode the phonetic transcription of spoken words by speaker dependent, but vocabulary independent phoneme models. The results show that the average number of final recognition errors will be over 15 % smaller, if the traditional K-means based initialization is substituted by SOM. The method described for fast SOM density approximation improves the total recognition time by over 40 % for the current online system compared to the default which uses independent complete searches for the best matching units.

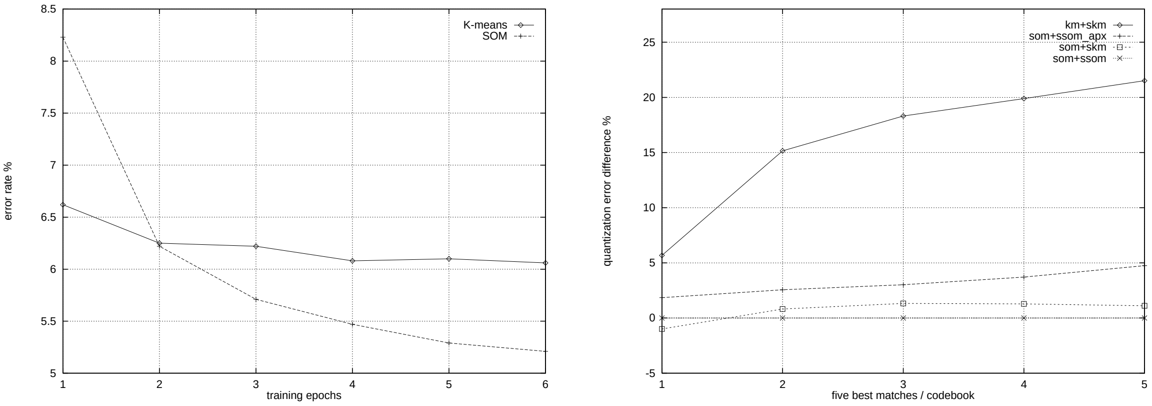
1 About the application

The automatic speech recognition is a classical example of a difficult pattern recognition problem. One widely applied solution is to assume the extracted sequence of features to be produced by Markov processes [10]. The model assumes that the system goes through a sequence of states to generate each phoneme and each state has slightly different probability density function for the signal features. Also the Markov property is assumed, i.e. the state transition probabilities are fully determined by the current state regardless of the history. In the decoding of the spoken phoneme sequences the essential problem is how to estimate the density functions in order to minimize the phoneme recognition errors. The best performance is expected to be achieved by aiming at the discrimination between the most probable path alternatives. For classification purposes the exact probability values are not important, but only the clear separation between the rival states. So instead of proper observation probabilities of the states, some more loosely defined likelihood values can be used. Also at each time instant, the high accuracy is normally needed only for the most likely states and so for the rest a rough approximation preserving only the correct order of magnitude is enough.

2 The learning process and methods

2.1 Smoothing of the parameters

Although the available training database seems to be large enough for practical purposes, due to its finiteness the portion of the data representing certain realizations of a certain classes can be insufficient for accurate modeling. For the total recognition accuracy these cases may have insignificant effect, but the density estimation should, anyway, be able to cope with very different amounts of training data for different states and still provide observation likelihoods in the right order of magnitude. For a class of features well presented in the data, it may still happen that after improper initialization, a part of the available modeling capability is left unreachable, because some of the Gaussian kernels will be practically unused. In the SOM type of learning the parameters of all the units can be drawn into use without extra cost by just starting the learning with wide initial neighborhood. With a less number of mixtures in the effective area the recognition accuracy can be even better at first, but when the training proceeds the quality of the result does not improve like it should (See Figure 1a).



a **b**

Figure 1: **a** The average test set error rates after successive HMM training epochs (by segmental LVQ3 [5]) on models initialized by K-means or SOM. On the average, the SOM initializations seem succeed better in initializing the parameters to assist in finding an optimal solution later. **b** The average quantization error of the five best matching units in the codebook corresponding to the phoneme label of the feature vector after the chosen segmental HMM training has been run for five epochs and the finer SOM topology from the initialization is lost. The notation 'km+skm' means that K-means was used in the initialization and segmental K-means in the further training. The 'ssom_apx' means testing by the approximative topological K-best search.

Another reason for the success of the SOM mixture densities is the smoothing of parameters as also reported to be vital for the discrete HMMs [9]. The need of smoothing arises from the finiteness of the training data producing in some cases too close adaptation to generalize properly. There are several methods that accomplish the smoothing as a separate process [10, 9], but by the SOM it is conveniently integrated to the learning process. By the SOM neighborhood adjustments each model parameter is affected by more input samples than would otherwise be possible and those additional samples are just those that naturally should fall into the range. This also offers another possible advantage by being able to train larger codebooks respective to the same available training material.

The size of the neighborhood controls the amount of smoothing and so, e.g., if the neighborhood is not allowed to decrease the fit to the data might also not be good enough. Compared to the codebooks trained without smoothing (e.g. trained by K-means) the accuracy for the best-matching Gaussian is usually worse, but the next (K-1)-best matches will be better, however. Figure 1b shows the relative difference of quantization errors for differently trained density codebooks. From the SOM initialized curves, the segmental K-means training provides the lowest 1-best quantization error. The curve for 'som+slvq' was removed from the figure, because it almost equals the 'som+skm' curve. One note from the Figure 1b is that the quality of the approximative search on the SOM codebook is rather close to the original, but decreases slowly when going further from the best match.

2.2 Segmental SOM for training the Gaussian MDHMMs

The initialization SOMs are trained using the basic version SOM algorithm [2], but the update rules for the MDHMMs apply the Batch Map [1]. So the mean vector $\boldsymbol{\mu}_{jm}$ for each Gaussian kernel $m = 1, \cdots, M_j$ of codebook j is set to the average of the associated set of feature vectors by:

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

where the indicator function $\delta(q_t, j) = 1$, q_t is connected to the codebook j (otherwise $\delta(q_t, j) = 0$). The state sequences $q = q_1, \cdots, q_T$ for the observation sequences $\boldsymbol{O} = \boldsymbol{O}_1, \cdots, \boldsymbol{O}_T$ are determined by the current segmentation. The neighborhood function $h_{o,m} > 0$, if the unit m belongs to the neighborhood of the best-matching unit o of the current codebook (thus $h_{o,m}$ depends on t via the index o). From the different neighborhood functions $h_{o,m}$ [2], the simple bubble type is used here. If individual covariance matrices would be needed for each Gaussian kernel the adaptation formula corresponding to (1) could be obtained by substituting $\boldsymbol{O}_t$ by matrix $(\boldsymbol{O}_t - \boldsymbol{\mu}_m)(\boldsymbol{O}_t - \boldsymbol{\mu}_m)^T$.

The mixture weight that connects an individual Gaussian kernel to a HMM state is set to reflect the contribution of the kernel for the output density of that state:

$$\hat{c}_{im} = \frac{\sum_{t=1}^{T} \delta(q_t, i) h_{o,m}}{\sum_{t=1}^{T} [\delta(q_t, i) \sum_{m=1}^{M_i} h_{o,m}]}, \tag{2}$$

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

The adaptation occurs iteratively so that the enhanced HMMs are used to provide an enhanced segmentation, which is again used to re-estimate the model parameters. In general, the batch adaptation gives faster convergence than the normal stochastic tuning and there is no need to worry how to set the learning rate. These properties are ideal for the mixture density estimation for HMMs. The total number of parameters to be estimated is huge and the accuracy improvement rate in the late iterations can be very slow so that a suboptimal solution must be accepted, in practice, as soon as it is good enough.

1. Use pre-segmented training data to initialize the density codebooks by training one SOM for each phoneme
2. Initialize the state dependent mixture weights according to the partitioning of the data obtained by the SOM units. If necessary the unit covariances (kernel widths) can be initialized likewise.
3. Segment all the training data (no pre-segmentation required) by Viterbi decoding of the phoneme sequences using the obtained HMMs.
4. Adapt all the parameters by a batch SOM iteration.
5. Repeat from step 3. until the upper limit for the number of iterations or the lower limit for the average change of parameters is reached.

The main difference to the conventional Viterbi training (by segmental K-means) is that here also the parameters of mixtures in the neighborhood of the best-match are adjusted. The tuning of the neighboring mixtures using partly the same feature vectors preserves their resemblance. As the radius of the applied neighborhood function decreases gradually to zero the fine structure of the topology is lost due to the folding of the map that strives for minimal distortion. The accuracy of the density estimation increases and thus improves also the recognition accuracy. Some coarse structure will still be available to provide the smoothing and the suggested faster search. The Figure 2 illustrates the damage of the topology when small neighborhood is applied.

The models trained by SOM are not optimized to discriminate between states of different models. For optimizing the classification boundaries the LVQ [2] can be applied to the sequence decoding to find the parts, where the models behave inappropriately and tune the densities towards the correct results (segmental LVQ3 [5]).

Like the SOM training with zero neighborhood, also the LVQ training causes the codebook to fold and loose the smoothness. The large scale structure of the codebook is not, however, entirely broken enabling still some potential for speed-up by density approximations.

3 Using the SOM topology

In addition of the accuracy, the density approximations must be very fast to compute, because the density values are required for each state of each model for all the observed feature vectors that typically come at about 0.01 s apart.

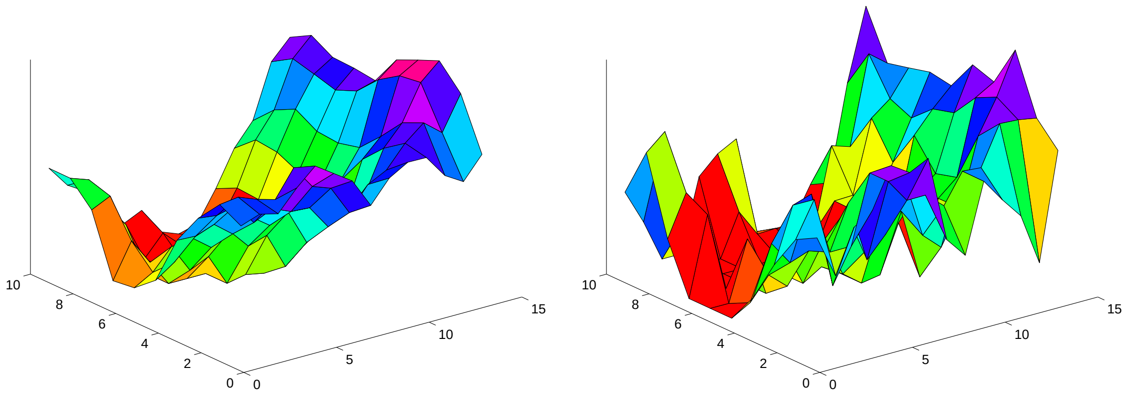


Figure 2: The responses of the individual mixture density components in the phoneme /A/ codebook organized into 10x14 grid are plotted for one randomly selected input vector. The first plot (left) is the situation when the radius is decreased to one and the second is after training with zero neighborhood

The short-time pdf estimate for a state is the sum of all associated Gaussian kernel densities weighted by the state specific mixture weights. However, in practice, a few best-matching kernels dominate the density estimate, so an approximation that excludes the other kernels can be applied. Since the search of the K best uses a significant part of the total computation load, some simple search speed-ups are may be very helpful. For example, if the initial K-best candidates can be "guessed" well, the sequential scanning in large codebooks can done much faster. If the approximate location of the K-best is known as well, the search area can be remarkably reduced assuming that the topology of the input space is preserved by the organization of the codebook units. If it is well preserved, most of the K-best matches in the SOM codebook are probably near each other in the SOM grid. The search area for K-best match can then be reduced to the (extended) neighborhood of some good best-match candidates [6].

3.1 Topological K-best search

The topological K-best search is presented here to point out one way to utilize the topology for a fast approximative search algorithm on large high-dimensional codebooks. In addition to the topological order, this method assumes also that the successive feature vectors usually resemble each other. Related techniques have been presented to speed up the winner search in the self-organization process [7, 3]. An important note is that the real best-matches are not necessarily found at every time instant, but rather a set of them that is related to the assumed best match. The presented method is expected to work well for the particular type of density estimation discussed in this work, but not for non-SOM codebooks or SOM applications in general.

Since the scanning of the density codebooks is done by using the partial distance computation (PDC) (i.e. the accumulation of vector component differences aborts when the reference distance is exceeded) the fastest execution time is expected, if the rejections of the non-winner mixtures can be determined as soon as possible. The components of the feature vectors should be operated in the order of decreasing influence, which coincides with decreasing variance order, if no special knowledge is available. E.g. for the cepstral coefficients of speech signal the natural order of the components is very close to that order.

By having as good as possible K-best ranked matches to compare the partial distances to, the number of the component checks of the poorly matching majority of the reference vectors and the changes in the selected vectors are minimized. For the stationary parts of a continuous signal like speech, the K-best from the search concerning the previous feature vector can be used to initialize the search. For the fast changing parts the previous statistics are less useful, but in the absence of other initializations they can still be applied.

For the topological search order described above, there is no way (except the complete search) to confirm that the local minimum found by expanding the neighborhood starting from the previous best match is also the global minimum of the codebook. Possible ways to gain more reliability would be to continue the search a little further, to use larger neighborhoods or to start the search simultaneously at different places. Here, anyhow, the first local minimum is accepted, i.e. the search is aborted when the match of the current unit is better than the match of any of its neighbors. The reason is that the local minimum is expected to often be an acceptable approximation and when it is not, as e.g. in some transitions parts, the change in features is probably so dramatic that the best solution would be a complete search. For freeing the codebook from a local minimum and tracking large feature changes the complete search is done periodically. The length of the period controls the accuracy and the speed of the search by altering the ability to react to fast changes.

1. Do the complete search periodically, but meanwhile approximate by the following:
2. Initialize the K-best by re-ranking the previous K-best matches.
3. Start the search from the neighbors of the best of those.
4. If a new best is found, search also its neighbors.
5. Repeat from step 4. until no more better matches are found.

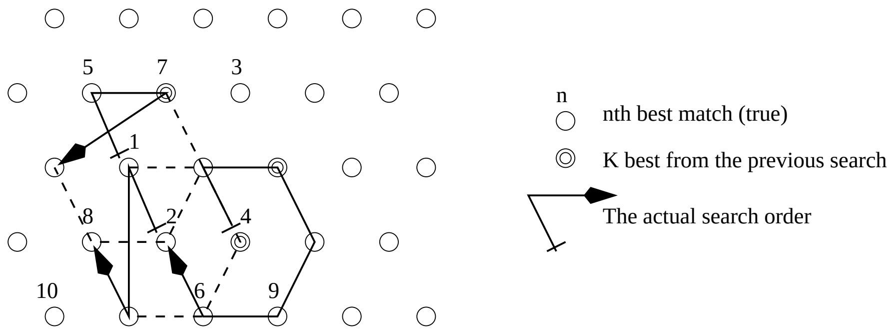


Figure 3: The topological search order for K-best search in SOM codebooks.

3.2 Other SOM-based methods for fast density approximation

The tree-search SOM [4] suits well to the fast approximative search for large codebooks, because the tree structure offers $O(\log N)$ search complexity. The possible loss of accuracy may follow from the sequential branching decisions by which most of the units are eliminated from the individual inspection. The neighboring search areas for tree-search SOM can be made slightly overlapping for better cover and search accuracy. For K-best search the effect of the branching decisions is softened by expanding the search into the lower layer search areas associated with the rival best-matches from the upper layer. For density approximation purpose the Gaussian kernels need to be trained only for the lowest SOM layer and the upper layers act only as a search tool.

The experimental results from the MDHMM recognition tests [6] indicate that the tree-search SOM can be used as a slightly worse performing, but a faster substitute to the normal SOM codebook. In the comparison by a ASR test to the corresponding normal SOM codebooks, the tree-search SOM in which the total recognition time decreased by 20 %, increased the average number of recognition errors by 14 %. However, the size of the mixture pool per phoneme was in these experiments rather small (121) to exploit fully the tree structure.

For a multi-codebook system like the phoneme-wise tied MDHMMs used here, the amount of necessary computations can be reduced by filtering out quickly the codebooks that are not close enough to the observation to have significance. A possible filtering criterion is presented and tested in [8, 6] based on the approximated best match and the width of the codebook.

4 Summary of experiments and results

Three separate repetitions of 350 Finnish words are used as the training material for each of the seven tested speakers. The reported error rate is the average number of phoneme errors (inserted, deleted and changed phonemes) made on a fourth separate speech set for each speaker. No post-processing is applied to take account of all phoneme errors introduced by the approximations.

The 80-dimensional feature vector is a concatenation of five sets of 15-component mel-cepstra and one energy component computed at different time intervals around the current 16 ms window moved forward in every 8 ms. The baseline MDHMM system [5] consists of a 5-state uni-directional HMM and a pool of 70 Gaussian mixtures of a fixed diagonal covariance matrix for each phoneme. The default performance figures after the SOM initialization and five epochs of the segmental SOM training are the average phoneme error rate of 5.9 % (5.3 % achieved by the segmental LVQ3) and the total recognition speed of 830 ms/word (currently increased to 290 ms/word by a new SGI O2 workstation) by using the PDC search for the 5-best search approximation.

As illustrated in Figure 1a, the average error rate can be decreased by about 17 % for the current configuration, by substituting the traditional K-means initialization by SOM [5]. The segmental SOM presented in this paper does not correspondingly improve the final error rate compared to segmental K-means or LVQ3, but the intention by maintaining the neighborhood longer is merely to study the later usability of the topology. For training significantly larger codebooks the segmental SOM might be important, anyhow.

Using the topological K-best search the average total recognition time decreases by 42 % and the recognition errors increase 10 % (from 5.9 to 6.5 %). If the segmental LVQ3 is used in the training, the increase of errors by the same search is not much higher (11 %), and by that way the 5.9 % error rate can be achieved at the speed of 480 ms/word.

5 Discussion

Compared to the rapid capacity improvements in the general-purpose workstations, the significance of the described speed-ups concerns only situations, where some system must be made to work online on a certain computer. The idea was mainly to show direct ways to exploit the ordered mapping generated automatically by the SOM, which does not depend as much on the data dimensionality or memory size as some traditional search methods do. Both the training by using the neighborhood smoothing and the topology based fast search methods are expected to scale well for larger dimensions and codebook sizes.

References

[1] T. Kohonen. Things you haven't heard about the Self-Organizing Map. In *Proceedings of ICNN-93*, pages 1147–1156, Piscataway, NJ, 1993. IEEE Service Center.

[2] T. Kohonen. *Self-Organizing Maps*. Springer, Berlin, 1995.

[3] T. Kohonen. The speedy SOM. Technical Report A33, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland, 1996.

[4] P. Koikkalainen and E. Oja. Self-organizing hierarchical feature maps. In *Proceedings of IJCNN-90*, volume II, pages 279–285, Piscataway, NJ, 1990. IEEE Service Center.

[5] 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.

[6] M. Kurimo and P. Somervuo. Using the Self-Organizing Map to speed up the probability density estimation for speech recognition with mixture density HMMs. In *Proceedings of ICSLP-96*, volume 1, pages 358–361, Philadelphia, PA, USA, October 1996.

[7] J. Lampinen and E. Oja. Fast self-organizing by the probing algorithm. In *Proceedings of IJCNN-89*, volume II, pages 503–507, Piscataway, NJ, 1989. IEEE Service Center.

[8] E. Lopez-Gonzalo and L. A. Hernandez-Gomez. Fast vector quantization using neural maps for CELP at 2400 bps. In *Proceedings of EUROSPEECH-93*, volume 1, pages 55–58, Berlin, Germany, September 1993.

[9] E. Monte. Smoothing HMMs by means of a SOM. In *Proceedings of ICSLP-92*, volume 1, pages 551–554, Banff, Canada, October 1992.

[10] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.