COMPETING HIDDEN MARKOV MODELS ON THE SELF-ORGANIZING MAP

Panu Somervuo

Helsinki University of Technology
Neural Networks Research Centre
P.O. Box 5400 FIN-02015 HUT
Finland
panu.somervuo@hut.fi

ABSTRACT

This paper presents an unsupervised segmentation method for feature sequences based on competitive-learning hidden Markov models. Models associated with the nodes of the Self-Organizing Map learn to become selective to the segments of temporal input sequences. Input sequences may have arbitrary lengths. Segment models emerge then on the map through an unsupervised learning process. The method was tested in speech recognition, where the performance of the emergent segment models was as good as the performance of the traditionally used linguistic speech segment models. The benefits of the proposed method are the use of unsupervised learning for obtaining the state models for temporal data and the convenient visualization of the state space on the two-dimensional map.

1. INTRODUCTION

The Self-Organizing Map (SOM) [5, 7] is an artificial neural network which defines a nonlinear transform from the input space to the set of nodes in the output space. Each node is associated with a model of the input space. Through an unsupervised learning process, the models become specially tuned and organized according to input patterns. The learning algorithm which leads to self-organization can be simplified into two steps [5, 7]. For each input sample:

1. Find the best-matching unit (BMU) on the map by using the chosen similarity measure.
2. Update the model of the BMU as well as the models of the neighboring units according to the current input sample.

These two steps are repeated during the training. The model updating in step 2. can be done incrementally after each input sample or in a batch process [7].

The SOM can be considered as a principle rather than one specific algorithm only. The input data in this work consists of unsegmented feature vector sequences with varying lengths. The goal is to develop an unsupervised segmentation method for data sequences using the competitive-learning principles of the SOM, and thereby to derive emergent segment models on the map. If the models associated with the SOM nodes are state models and the similarities between the models and data are computed using probabilities, this leads to the concept of hidden Markov models (HMMs) [10]. Each data segment of the sequence can then be interpreted as an output of one map node.

This paper proceeds by first discussing HMMs, then presenting the learning scheme for competitive HMMs on the SOM, and then giving the results of the experiments with speech.

2. HIDDEN MARKOV MODELS IN SPEECH RECOGNITION

Hidden Markov models (HMMs) are models for sequential, stochastic data. They are commonly used in speech recognition [10] where their benefit is to tie separate observations in time together and so utilize the mutual time-dependency and order of acoustic phenomena in recognition. The speech patterns are

This work was supported by Nokia Foundation.

compactly represented as a state network. Besides speech recognition, HMMs are applicable to various other tasks too, e.g., they have been used in natural handwriting recognition, text analysis, coding theory, ecology, and molecular biology.

The states and their interactions are the core of the HMM. Formally the HMM is defined as a triple

$$\lambda = (A, B, \boldsymbol{\pi}), \tag{1}$$

where $A = [a_{ij}]$ is an $N \times N$ matrix of the state transition probabilities, $B = \{b_i\}_{i=1}^{N}$ is a set of observation probability densities of N states, and $\boldsymbol{\pi}$ is an initial state probability vector.

There exist well-known optimization algorithms for the final parameter estimation task relying on different criteria such as maximum-likelihood (ML) principle, mutual information, or error-correctiveness. However, a fundamental question before any parameter estimation algorithm can be applied, is how to define and parametrize an HMM state or a state sequence. Usually prior knowledge is required for defining the sequence units to be modeled or deciding what kind of representations the states should have. There should be a balance between the specificity of the models and the amount of the training data. It is clear that this is not the problem of speech recognition only, but one of the main problems of modeling in general.

In speech recognition, the traditionally modeled speech units are whole words or linguistic subwords like phonemes or phoneme pairs. True whole-word models require training utterances for each word to be recognized. Thus more training data can be used for each model if all words can be constructed using a limited set of building blocks. A whole-word model can be constructed using the building blocks by means of a pronunciation dictionary. Usually the subword units are phonemes and the pronunciation of the word is represented as a phoneme sequence.

If the number of the subword models is small, there is more training data available for each model but the acoustic variation inside each model may be large. The realizations of the phonemes, the phones, behave differently depending on their acoustic context. The preceding phones have effect on the ensuing ones and vice versa. Thus more robustness can be obtained in the modeling by taking the local context of the phones into account. Models can be constructed for larger linguistic speech units such as phoneme pairs or triples. The cost of modeling concatenated phonemes is, however, the increased number of models and thus the need for larger amount of training data. Since the amount of training data is usually limited, some kind of parameter tying is then required.

The idea in the present work is that the HMMs are not trained for pre-defined linguistic segments, but the SOM will find the segments from the feature sequences in an unsupervised manner. The resulting models may then be generalized phones, diphones or triphones, or they may represent smaller and more detailed parts of the phones. The specificity and characteristics of the models are determined by the properties of the input data.

Earlier experiments with the data-driven segment units have been presented in [9, 3, 4]. Generative topographic mapping (GTM) [2, 1] also has resemblance to the current work.

3. COMPETITIVE LEARNING OF HIDDEN MARKOV MODELS

In principle, the models associated with the SOM nodes can be chosen arbitrarily. They do not even have to be equally parametrized since the competition of the models is based on their activity to response to the input patterns. The model adaptation means just tuning the models to better fit to the input [6], and this can be carried out differently depending on the parametrization of the model.

In this work the models associated with the SOM nodes are single- or multi-state Markov models. The speech segments to be modeled are not pre-defined; the training is totally unsupervised. Besides using the SOM as a framework for training emergent state models, the SOM is also used in the present work for estimating observation pdfs of the states; SOM-based vector-codebooks are used as the basis of the pdfs and each codebook vector is one kernel mean of the Gaussian mixture [8].

Hidden Markov models are usually trained by using the Baum-Welch or the segmental k-means algorithm [10]. The Baum-Welch algorithm adapts the state models of all possible state sequences according to their likelihood to produce the input whereas the segmental k-means algorithm adapts only the models of one best state sequence. The single best state sequence is obtained by using the Viterbi algorithm [10]. Selecting only the best sequence corresponds to the winner-take-all principle of the SOM. The Viterbi algorithm is used in the present work for choosing the best-matching model sequence. The smoothness of the training and the sharing of the training data between models is then provided by the neighborhood of the SOM.

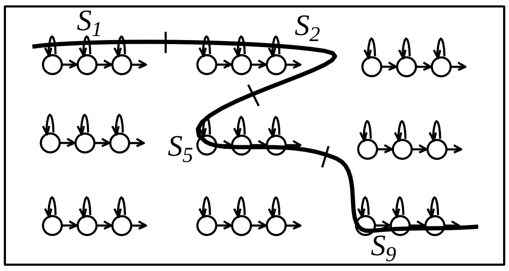


Figure 1: *Competitive-learning segment models on the SOM. Each map node is associated with an HMM having one or more states (a three-state left-to-right model in this example). The thick line represents the segmentation of an input sequence. The data segments are denoted by S_i, where i is a BMU index. The models of BMUs and neighboring units are then updated by the corresponding segments.*

The training begins by initializing the state models. Random initialization can be used, but more controlled initialization is obtained by first training a large vector-SOM and then partitioning it into clusters. Each cluster is then considered as a state. Each vector of one cluster becomes the kernel mean of the pdf and it is provided with a weight coefficient. The training proceeds by repeating the following two steps a sufficient number of times: for each input sequence in the training set,

1. *BMU-search.* Segment the input sequence to the map units by the Viterbi algorithm.
2. *Model adaptation.* Use each data segment for updating the model of its BMU and the models of the neighboring nodes.

The BMU-search, which means the same as the segmentation in case of feature vector sequences, is illustrated in Fig. 1. Due to the Viterbi algorithm, the individual feature vectors in the BMU-search are not matched against map nodes as separate items but as a part of the whole feature sequence. The Viterbi algorithm finds an optimal state sequence fitting the data sequence in the HMMs. For the observation vector sequence $\boldsymbol{O}_1^T = [\boldsymbol{O}_1 \boldsymbol{O}_2 \dots \boldsymbol{O}_T]$, this is

$$\hat{q}_1^T = \arg\max_{q_1^T} \boldsymbol{\pi}_{q_0} \prod_{t=1}^{T} a_{q_{t-1}q_t} b_{q_t}(\boldsymbol{O}_t), \tag{2}$$

where q_t is the state index and t is the time index.

The pdf of the state i is modeled by the Gaussian mixture with M_i kernels:

$$b_i(\boldsymbol{O}_t) = \sum_{m=1}^{M_i} c_{im} \mathcal{N}(\boldsymbol{O}_t; \boldsymbol{\mu}_{im}, \boldsymbol{\Sigma}_{im}), \tag{3}$$

where c_{im} is the mth mixture coefficient and $\mathcal{N}$ is the Gaussian kernel with mean vector $\boldsymbol{\mu}_{im}$ and covariance matrix $\boldsymbol{\Sigma}_{im}$. Each kernel mean is one prototype vector of the vector-SOM.

The neighborhood function h is used on the state-SOM when updating the state models. Another neighborhood function h' is used inside each state for the vector-SOM when updating the kernel means. The neighborhood function controls the learning rates of all map units and the organization of the whole map. It may have, e.g., the shape of a Gaussian.

For single-state models on the SOM, the new values of the parameters after batch adaptation are

$$\bar{c}_{im} = \frac{\sum_{t=1}^{T} h(r_{\hat{q}_t}, r_i) h'(r'_{w_i(\boldsymbol{O}_t)}, r'_{im})}{\sum_{t=1}^{T} h(r_{\hat{q}_t}, r_i) \sum_{k=1}^{M_i} h'(r'_{w_i(\boldsymbol{O}_t)}, r'_{ik})}, \tag{4}$$

$$\bar{\boldsymbol{\mu}}_{im} = \frac{\sum_{t=1}^{T} h(r_{\hat{q}_t}, r_i) h'(r'_{w_i(\boldsymbol{O}_t)}, r'_{im}) \boldsymbol{O}_t}{\sum_{t=1}^{T} h(r_{\hat{q}_t}, r_i) h'(r'_{w_i(\boldsymbol{O}_t)}, r'_{im})}, \tag{5}$$

$$\bar{\boldsymbol{\Sigma}}_{im} = \frac{\sum_{t=1}^{T} h(r_{\hat{q}_t}, r_i) h'(r'_{w_i(\boldsymbol{O}_t)}, r'_{im})(\boldsymbol{O}_t - \boldsymbol{\mu}_{im})(\boldsymbol{O}_t - \boldsymbol{\mu}_{im})^\top}{\sum_{t=1}^{T} h(r_{\hat{q}_t}, r_i) h'(r'_{w_i(\boldsymbol{O}_t)}, r'_{im})}, \tag{6}$$

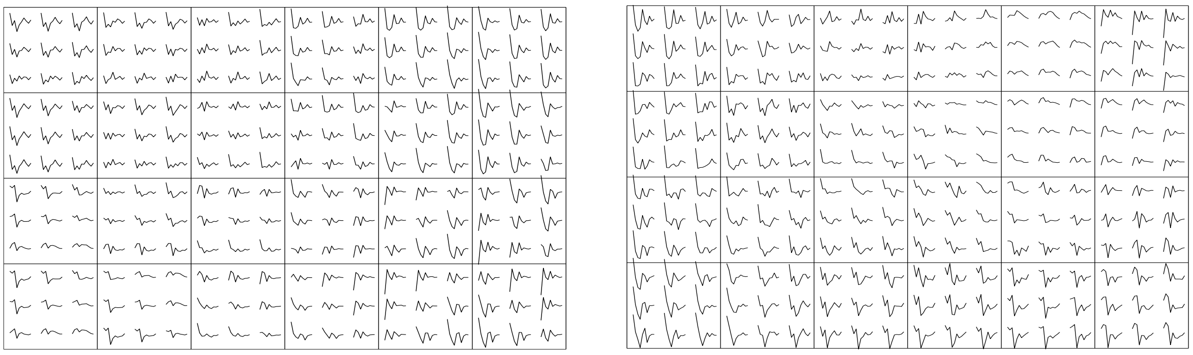


Figure 2: *Two examples of 6×4-unit state-SOMs. Each state is modeled by a 3×3-unit vector-SOM and a Gaussian kernel with a fixed width is attached to each vector. 10-dimensional kernel means are illustrated inside the states. The SOM on the left was initialized with random values for kernel means and the SOM on the right was initialized by partitioning an 18×12-unit vector-SOM into 3×3-unit blocks which then became the states.*

where $r_{\hat{q}_t}$ and r_i are the position vectors of the states $\hat{q}_t$ and i on the state-SOM, respectively, and $r'_{w_i(\boldsymbol{O}_t)}$ and r'_{im} are the position vectors of the code vectors $w_i(\boldsymbol{O}_t)$ and im on the vector-SOM, respectively. Index $w_i(\boldsymbol{O}_t)$ denotes the best-matching unit of codebook i for observation vector $\boldsymbol{O}_t$.

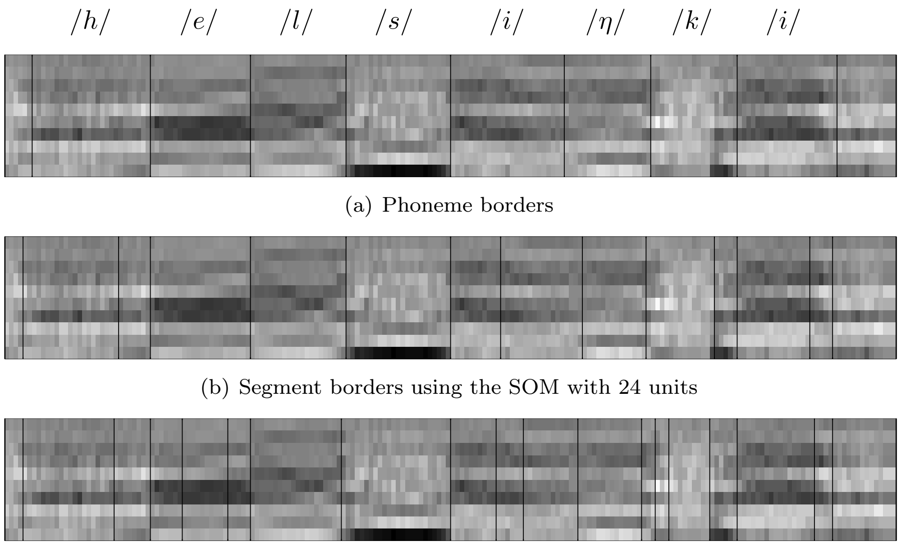
If the state models are randomly initialized, some of them may get large segments of the input in the beginning of the training and thus become dominating. The transition probabilities back to the dominating states are then likely to become large and these models do not become selective. This is avoided if the transition probabilities between the states are set equal in the beginning of the training and after the state-SOM neighborhood has been shrunk to small value, they are adapted. This procedure was found to give good results in the experiments. Another way to spread the states into the state space is to use the vector-SOM initialization. This means that the state models are first trained for static vectors before the modeling of the dynamics of the input takes place.

If multi-state HMMs are used, the individual models of the nodes must learn the time structure of the input. Using randomly initialized models, the parameters of the states inside each HMM can at first be tied and then loosened during the training so that the state sequences inside HMMs will gradually adapt to the time structure of the segments. This suggests using the neighborhood over the states inside each HMM which controls the stiffness of the time resolution. Another approach, which was used in the present work, is to first train single-state models and then split these HMMs into left-to-right models. The training proceeds by segmenting the training sequences, and for each segment, the Viterbi algorithm is used separately for each HMM on the map for partitioning the data into the states. The neighborhood function h is used for controlling the learning rate and the centroid of h is always in the BMU of the data segment.

Each individual HMM models the dynamics of only one segment of the input. It is therefore important that all HMMs are connected to each other. These connections associated with the transition probabilities join the local segment models together into one network giving the possibility to model whole input sequences.

4. EXPERIMENTS

Fig. 2 illustrates two examples of the state-SOMs. The sizes of the maps are 6×4 units. 10-dimensional cepstrum vectors were computed from Finnish speech. An HMM consisting of one state was associated with each map node. Each state was modeled by a mixture of 9 Gaussians. This was implemented by using a 3×3-unit vector-SOM. In the first experiment, the initial values of kernel mean vectors were set randomly. In the second experiment, the initial values of kernel means were obtained by training an 18×12-unit vector-SOM and then partitioning it into 3×3-unit blocks. The widths of the Gaussians were fixed. The transition probabilities between states were uniform in the beginning of the training and they were adapted only during two last epochs of the training. One training epoch consisted of segmenting all training sequences and then updating the parameters of the models in a batch. As a result of using the neighborhood in the training, similar states are located near each other on the map. By visually evaluating the similarity of the states, however, it must be noted that one state behaves as an entity. The order of the kernels needs not be the same in the neighboring states.



(a) Phoneme borders

(b) Segment borders using the SOM with 24 units

(c) Segment borders using the SOM with 216 units

Figure 3: *Segmentation of word utterance 'HELSINKI'. 10-dimensional feature vectors are represented as column vectors on the horizontal time-axis.*

An interesting issue is how to evaluate and interpret the nature of the speech segments corresponding to the models on the SOM. The states can be labeled according to the linguistic speech units. The result of the segmentation can also be visualized by plotting the feature sequences with segment borders. Another possibility is to compare the likelihood of the models to produce the data. It is also of interest to investigate whether the derived segment models are relevant and consistent for the speech recognition task. For this purpose, a word recognition experiment was conducted.

The training set consisted of 350 Finnish words which were uttered three times by a male speaker. In the test set the same 350 words were uttered the fourth time. The vocabulary was 340 words. 10-dimensional cepstrum vectors were used as features. An 18×12-unit vector-SOM was first trained. This vector-SOM was used for initializing three state-SOMs. In the first state-SOM each state was modeled by only one Gaussian. There were thus 18×12 states on the map. When the large vector-SOM was partitioned into 2×2-unit and 3×3-unit blocks, this gave the initializations for 9×6-unit and 6×4-unit state-SOMs, respectively. Diagonal covariance matrices were used with Gaussian kernels. The single-state HMMs were expanded into three-state left-to-right models and they were trained by unsupervised learning. When the training data was segmented to the models after the training, the average durations of one segment were 8.7, 9.0, and 10.2 frames (70, 72, and 81 ms) for 216-, 54-, and 24-unit SOMs, respectively. An example of the segmentation is shown in Fig. 3.

The reference word models were obtained by computing the best state sequence of each word utterance of the training set by using the Viterbi algorithm. These state sequences constituted the basis of the word models and they were stored in the pronunciation dictionary. When forming the word models, the durations of the states were compressed so that the same state was not allowed to follow itself more than a limited number of times. This is denoted by 'max segment length in word model' in Table 1. The word recognition was then performed using the test data. Comparing the results of the three maps (see Table 1), the best time resolution of the largest SOM (216 competing models) gave the best word recognition result.

For comparison, the word recognition was experimented also using linguistic speech units. Left-to-right HMMs were trained for 20 phonemes. One additional model was trained for silence. The reference word models were constructed by concatenating the phoneme models according to the correct pronunciation of the word. The best recognition result was then 99.7 per cent. This is very close to the results obtained by using segment models derived in an unsupervised manner.

5. CONCLUSION

The SOM gives a framework for training emergent models using unsupervised learning. The central idea in the present work was to associate state models with the SOM nodes. Using the probability as a similarity measure between the models and data led to the concept of hidden Markov models as the

Table 1: *Speech recognition using segment models derived by unsupervised learning. Each segment model was a three-state HMM. The vocabulary consisted of 340 Finnish words. Number of correct words per cent.*

number	max segment length in word model				
of HMMs	1	2	3	4	5
6×4	97.7	97.4	96.6	97.7	97.4
9×6	97.1	97.7	98.3	98.6	98.0
18×12	98.0	99.1	99.7	100.0	100.0

nodes. The Viterbi algorithm was embedded in the SOM training and this provided an unsupervised segmentation method for sequential data. Usually the training of the HMMs is supervised which requires that the segment units to be modeled are pre-defined.

A state of the HMM is characterized by an observation probability density function which is usually represented as a mixture density. Since the kernel means of the mixture densities are feature vectors, the states can be initialized by first training a large vector-SOM. Each cluster of vectors is then considered as a state. A continuous-valued pdf is obtained when the quantization error between data vectors and kernel means is utilized. Gaussian kernels were used in the present work.

The proposed method was tested using speech data. The speech segment models derived by unsupervised learning were evaluated in the word recognition task. The results were comparable with the best results obtained by using conventional linguistic speech unit models which had been trained in an supervised manner. A two-dimensional SOM array offered also a convenient way for visualizing the state space.

6. REFERENCES

[1] C. Bishop, G. Hinton, and I. Strachan, "GTM through time", *Proc. IEE ICANN-97*, pp. 111-116, Cambridge, 1997.

[2] C. Bishop, M. Svensén, and C. Williams, "GTM: the generative topographic mapping", *Neural Computation*, vol. 10, no. 1, pp. 215-234, 1998.

[3] C. Lee, F. Soong, and B. Juang, "A segment model based approach to speech recognition", *Proc. ICASSP-88*, pp. 501-504, New York, April 1988.

[4] C. Lee, B. Juang, F. Soong, and L. Rabiner, "Word recognition using whole word and subword models", *Proc. ICASSP-89*, pp. 683-686, Glasgow, May 1989.

[5] T. Kohonen, "Self-organized formation of topologically correct feature maps", *Biological Cybernetics*, vol. 43, pp. 59-69, 1982.

[6] T. Kohonen, "Generalizations of the Self-Organizing Map", *Proc. IJCNN-93* (IEEE Service Center, Piscataway, NJ 1993), pp. I-457-462, Nagoya, 1993.

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

[8] M. Kurimo, "Using Self-Organizing Maps and Learning Vector Quantization for Mixture Density Hidden Markov Models", *PhD thesis*, Helsinki University of Technology, Espoo, Finland, 1997.

[9] R. Nag, S. Austin, and F. Fallside, "Using hidden Markov models to define linguistic units", *Proc. ICASSP-86*, pp. 2239-2242, Tokyo, 1986.

[10] L. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition", *Proc. IEEE*, vol. 77, no. 2, pp. 257-286, February 1989.