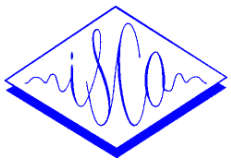

2nd International Conference on
Spoken Language Processing (ICSLP 92)
Banff, Alberta, Canada
October 12-16, 1992

TOPOLOGY PRESERVATION FOR SPEECH RECOGNITION

Gregory R. De Haan

Speech Technology Laboratory
Div. of Panasonic Technologies, Inc.
3888 State Street, Suite 202
Santa Barbara, CA 93105

Ömer Eğecioğlu

Department of Computer Science
University of California, Santa Barbara
Santa Barbara, CA 93106

ABSTRACT

When comparing Self-Organizing Feature Map (SOFM) learning with the generalized Lloyd algorithm for vector quantizer design, it becomes apparent that an implicit goal of SOFM learning is to reduce the summed distortion between the input vector and all vectors in the neighborhood of the closest SOFM weight vector. We present appropriate neighborhood-based distortion measures for SOFM and derive a variant of the generalized Lloyd algorithm which, unlike traditional SOFM learning, monotonically reduces neighborhood distortion. We show that topology preservation naturally results from minimization of neighborhood distortion, and such distortion can be used to quantify the degree to which topology is preserved in SOFMs. We also report on the use of the topology preserving properties of SOFMs for speaker independent isolated digit recognition. SOFMs are found to be effective for input pattern normalization, pattern recognition, and feature integration.

1 INTRODUCTION

From the late 1970's to the present, Teuvo Kohonen has led the development of the popular Self-Organizing Feature Map (SOFM) neural networks. SOFM were found to have many intriguing capabilities [15]: they can preserve the topological relationships in a multi-dimensional space, while performing data reduction (*topology preservation*); they can represent those dimensions of the input space with high variance (*automatic selection of feature dimensions*); they can represent hierarchically related data on a single layer of processing units, instead of using separate layers to represent different levels of abstraction; they can be trained quickly and were successfully applied to a practical problem– automatic speech recognition[14].

The update rule for SOFM is a pattern learning [20] version of the generalized Lloyd algorithm for vector quantizer design [6], where an SOFM unit's weight vector corresponds to a vector quantization codevector. SOFM training adds a neighborhood mechanism which gives them their additional capabilities. Like vector quantizers, SOFM can produce good codebooks. Unlike vector quantizers, SOFM can order the codevectors such that their position on the SOFM encodes information about the topological relationships between the codevectors. The other properties described above are by-products of this topology preserving ordering.

SOFM are trained to achieve topology preservation by first using large neighborhoods to obtain a reasonable ordering of vectors, and then reducing the neighborhoods to obtain good codebooks in a vector quantization sense: i.e., by minimizing the expected distortion introduced by mapping an input vector to a codevector. Unfortunately, however, poor ordering of these vectors can easily occur [8][12]. While it is easy to detect poor ordering in simple "toy" problems, it has proven difficult to determine the quality of the ordering with complex multidimensional inputs, like speech spectra. Only recently have objective measures of how well the vectors are ordered been introduced [1][2][4][5][16].

Herein neighborhood distortion functions (NDF) are used to quantify the degree of topology preservation in an SOFM. With NDF, all codevectors in the neighborhood of the best matching codevector contribute to the distortion. NDF are an effective measure of topology preservation because lowering the distortion encourages the preservation of both adjacency and non-adjacency: distortion is lower when the closest unit and its neighbors have similar weight vectors, and distortion is higher when the unit and its neighbors have dissimilar weight vectors.

We are studying the special capabilities of SOFM for use in speech recognition systems. Instead of assigning labels to the units, as in [14], our recognizers use the position on the SOFM of the units which respond. In contrast, the approach taken in [14] does not utilize the topology preserving ordering produced by SOFM (or its by-products). Torkkola, Kokkonen, and Kangas have independently studied the direct use of topology preservation for speech recognition [19][11].

2 SOFM LEARNING

2.1 Traditional SOFM Learning

An SOFM consists of neuron-like units in a d-dimensional space: typically $d = 2$, in analogy to the (approximately) 2-dimensional structure of human cortex. Usually the units are placed uniformly on the map, with either a rectangular or hexagonal tessellation. The map is trained iteratively, with each unit u_i updating its weight vector w_i. For each iteration, the traditional SOFM learning algorithm performs the following:

1. Find the unit u whose weight vector is closest to the input vector.
2. Update the weight vectors of unit u and all units in the neighborhood of u as a convex combination of the input vector and the corresponding weight vector, i.e.

$$w_j(t+1) = \begin{cases} (1-\alpha_t)w_j(t) + \alpha_t x & \text{if } u_j \in \text{neighborhood } (u), \\ w_j(t) & \text{otherwise.} \end{cases}$$

Note that the best matching unit is chosen by assuming that the nearest neighbor rule [6] holds. Also, when an input is presented, a given unit u is updated if and only if the closest unit is in u's neighborhood; thus the expected value of a unit's weight vector is the *centroid* of the union of the Voronoi regions corresponding to the unit and its neighbors [13]. Therefore traditional SOFM learning appears to extend the generalized Lloyd algorithm to account for the neighborhood mechanism. Based on these implicit assumptions of Kohonen learning, we have developed a batch training algorithm for SOFM which makes these assumptions more explicit. For each pass through the training data, this batch algorithm does the following:

1. Partition: assign each input vector to the unit with the closest weight vector.
2. Update: using the partition from step 1, update the weight vector for each unit u to be the centroid of the set of input vectors assigned to either u or any of the neighbors of u.

While this batch algorithm appears to be a simple extension of the generalized Lloyd algorithm, it has an important deficiency when the neighborhood includes more than just the best matching unit, i.e., the neighborhood radius $r > 0$. Upon running experiments with the batch algorithm and $r > 0$, we found that the neighborhood distortion, as defined in Section 2.2.2, was *not* monotonically decreased with each pass through the training data. We therefore performed an analysis of SOFM which minimize neighborhood distortion functions [4]. For the simple case of a 1-D SOFM with 1-D inputs and $r \geq 1$, neither the partitioning nor the centroid assumptions of the batch algorithm (and Kohonen's algorithm) held for optimal codebooks. Even for Voronoi SOFM, where we enforce the nearest neighbor rule, optimal codebooks violated the centroid assumption.

Please note that here we have assumed that every unit in the neighborhood is treated equally. While this is typically the case in the literature, graded neighborhood functions have been used. These algorithms can easily incorporate such graded neighborhood functions: for the pattern learning version, simply multiply the learning rate by the neighborhood value for the unit given the closest unit; for the batch version, the update step uses a weighted combination of the respective centroids.

2.2 Neighborhood Distortion-Based Learning

Upon inspection of traditional SOFM learning and its relationship to vector quantization, it is apparent that traditional learning implicitly uses a distortion measure which supports the neighborhood mechanism. Such a neighborhood distortion measure should include contributions from each unit in the neighborhood of the best matching unit. In this section we present a formulation for *weighted* vector quantization which extends the GLA to account for the SOFM neighborhood mechanism. A learning algorithm is then derived which monotonically reduces the neighborhood distortion summed over the training set.

2.2.1 Weighted Vector Quantization (WVQ)

The objective of WVQ is to construct codebooks which minimize the expected value of a "multiple-vector" distortion function. We consider distortion functions which are a weighted sum of the distortion between the input vector and each of the codevectors. Since we did not want to make any restrictions on the neighborhood distortion functions for SOFM, we did not place any significant constraints on the weights used in the distortion function. Nonetheless, the discerning reader will notice the similarities between WVQ and VQ in the presence of discrete memoryless channel noise [21].

Associated with a weighted vector quantizer is a finite set of N codevectors $Y = \{y_1, y_2, \ldots, y_N\} \subset \Re^n$; an $N \times N$ weight matrix of real numbers $\mathbf{W}$, an N-region partition $R_1, R_2, \ldots, R_N$ of $\Re^n$, where R_i is the region associated with codevector y_i; an encoder function $\mathcal{C} : \Re^n \rightarrow [\mathrm{N}]$, where $[\mathrm{N}] = \{1, 2, \ldots, N\}$ which takes the input $x \in \Re^n$ and yields the index i of the region R_i, i.e. $\mathcal{C}(x) = i \longleftrightarrow x \in R_i$; and a decoder function $\mathcal{W} : [\mathrm{N}] \rightarrow \Re^N$, defined below.

Here a weighted vector quantizer maps an input $x \subset \Re^n$ to a weight vector, as follows:

$$\mathcal{W}(\mathcal{C}(x)) = (\mathbf{W}_{\mathcal{C}(x),1}, \mathbf{W}_{\mathcal{C}(x),2}, \ldots, \mathbf{W}_{\mathcal{C}(x),N}), \tag{1}$$

where the first component of the resultant vector corresponds to codevector y_1, and the second component corresponding to y_2, and so on. Thus a weighted vector quantizer maps an input x to row $\mathcal{C}(x)$ of the matrix $\mathbf{W}$.

Given N and the weight matrix $\mathbf{W}$, the performance of a weighted-vector quantizer is measured by the average distortion

$$\mathcal{D}(\mathcal{W}) = E\left[\Delta(x, \mathcal{C}(x))\right],$$

where $\Delta : \Re^n \times [\mathrm{N}] \rightarrow \Re$, the weighted vector quantization distortion function, is defined as

$$\Delta(x, \mathcal{C}(x)) = \sum_{j=1}^{N} \delta(x, y_j) \mathbf{W}_{\mathcal{C}(x),j}. \tag{2}$$

where $\delta : \Re^n \times \Re^n \rightarrow \Re$ is a distance function.

Given N, $\mathbf{W}$, and the input distribution, a weighted vector quantizer $\mathcal{W}'$ is optimal if, for any weighted vector quantizer $\mathcal{W}$,

$$\mathcal{D}(\mathcal{W}') \leq \mathcal{D}(\mathcal{W})$$

WVQ Optimality Conditions

Next we present two necessary conditions for optimal WVQ, which are natural extensions of the Nearest Neighbor Partitioning and the Centroid Conditions of vector quantization [6].

First we consider necessary conditions for optimal WVQ partitioning. Given N, $\mathbf{W}$, the input distribution and the set of codevectors Y, we want to obtain the optimal partitioning of the input space to minimize neighborhood distortion.

The WVQ Partitioning Condition: For any optimal weighted vector quantizer, the partition is as follows:

$$R_i \supset \{x \in \Re^n : \Delta(x, i) < \Delta(x, j), \text{ for all } i, j \in [\mathrm{N}], i \neq j\} \tag{3}$$

PROOF

$$\begin{aligned} E\left[\Delta(x, \mathcal{C}(x))\right] &= \sum_{i=1}^{N} E\left[\Delta(x, i) | x \in R_i\right] Pr\left[x \in R_i\right] \\ &= \sum_{i=1}^{N} E\left[\sum_{j=1}^{N} \delta(y_j, x) \mathbf{W}_{ij} | x \in R_i\right] Pr\left[x \in R_i\right] \\ &= \sum_{i=1}^{N} \left[\left(\sum_{j=1}^{N} \int_{x \in R_i} \delta(y_j, x) \mathbf{W}_{ij} dx\right) Pr\left[x \in R_i\right]\right] \end{aligned}$$

Note that

$$\begin{aligned} &\sum_{i=1}^{N} \left[\left(\sum_{j=1}^{N} \int_{x \in R_i} \delta(y_j, x) \mathbf{W}_{ij} dx\right) Pr\left[x \in R_i\right]\right] \\ &\geq \sum_{i=1}^{N} \left[\min_{1 \leq k \leq N} \left(\sum_{j=1}^{N} \int_{x \in R_i} \left[\delta(y_k, x) \mathbf{W}_{kj}\right] dx\right) Pr\left[x \in R_i\right]\right] \end{aligned}$$

and thus distortion is minimized if and only if the partitioning ensures that equality holds. Equality occurs only if (3) holds. □

Next we give a necessary condition for optimal WVQ codevector placement. Given N, $\mathbf{W}$, the input distribution and the partition, we want to obtain the optimal values of the codevectors Y to minimize neighborhood distortion.

The WVQ Codevector Placement Condition: For any optimal weighted vector quantizer where the distance function δ is squared Euclidean distance, i.e. $\delta(x, y) = \|x - y\|^2$, the codevectors are given by

$$y_i = \sum_{j=1}^{N} \frac{\mathbf{W}_{ji} Pr[x \in R_j]}{\sum_{k=1}^{N} \mathbf{W}_{ki} Pr[x \in R_k]} E[x \mid x \in R_j]. \tag{4}$$

Since $E[x \mid x \in R_j]$ is the centroid of the region R_j, optimal codevectors are placed at a linear combination of the centroids of the N regions in the partition. If $\mathbf{W}_{ij} \geq 0$ for all $i, j \in [\mathrm{N}]$, then (4) implies that optimal codevectors are placed at a convex combination of the centroids.

PROOF

$$\begin{aligned} E[\Delta(x, \mathcal{C}(x))] &= E[\sum_{i=1}^{N} \delta(x, y_i) \mathbf{W}_{\mathcal{C}(x),i}] \\ &= \sum_{i=1}^{N} \sum_{j=1}^{N} (E[\delta(x, y_i) \mathbf{W}_{ji} \mid x \in R_j] \; Pr[x \in R_j]) \\ &= \sum_{i=1}^{N} \sum_{j=1}^{N} (\mathbf{W}_{ji} E[\delta(x, y_i) \mid x \in R_j] \; Pr[x \in R_j]) \\ &= \sum_{i=1}^{N} \sum_{j=1}^{N} (\mathbf{W}_{ji} Pr[x \in R_j]) \, E[\delta(x, y_i) \mid x \in R_j] \end{aligned}$$

Now consider the inner sum with the squared Euclidean distance distortion, and let $F_j = \mathbf{W}_{ji} Pr[x \in R_j]$.

$$\begin{aligned} &\sum_{j=1}^{N} F_j E\left[\|y_i - x\|^2 \mid x \in R_j\right] \\ &= \sum_{j=1}^{N} F_j E\left[\|x\|^2 | x \in R_j\right] + \sum_{j=1}^{N} F_j E\left[\|y_i\|^2\right] - 2 \sum_{j=1}^{N} F_j E\left[x \cdot y_i \mid x \in R_j\right] \\ &= \sum_{j=1}^{N} F_j E\left[\|x\|^2 | x \in R_j\right] + \sum_{j=1}^{N} F_j \|y_i\|^2 - 2 \sum_{j=1}^{N} F_j \left(y_i \cdot E\left[x \mid x \in R_j\right]\right) \\ &= \sum_{j=1}^{N} F_j E\left[\|x\|^2 | x \in R_j\right] + \sum_{j=1}^{N} F_j \left(\|y_i\|^2 - 2(y_i \cdot E\left[x \mid x \in R_j\right])\right) \end{aligned}$$

Since

$$\|y_i - E[x \mid x \in R_j]\|^2 = \|y_i\|^2 + \|E[x \mid x \in R_j]\|^2 - 2(y_i \cdot E[x \mid x \in R_j])$$

we have

$$\begin{aligned} &\sum_{j=1}^{N} F_j \left(\|y_i\|^2 - 2(y_i \cdot E\left[x \mid x \in R_j\right])\right) \\ &= \sum_{j=1}^{N} F_j \left(\| y_i - E[x \mid x \in R_j] \|^2 - E[x \mid x \in R_j]\right) \\ &= \sum_{j=1}^{N} F_j \| y_i - E[x \mid x \in R_j] \|^2 - \sum_{j=1}^{N} E[x \mid x \in R_j] \end{aligned}$$

Therefore the codevector y_i must be placed so as to minimize

$$\sum_{j=1}^{N} F_j \| y_i - E[x \mid x \in R_j] \|^2 \tag{5}$$

and thus

$$\begin{aligned} y_i &= \frac{\sum_{j=1}^{N} F_j E[x \mid x \in R_j]}{\sum_{j=1}^{N} F_j} \\ &= \sum_{j=1}^{N} \frac{F_j}{\sum_{k=1}^{N} F_k} E[x \mid x \in R_j] \\ &= \sum_{j=1}^{N} \frac{\mathbf{W}_{ji} Pr[x \in R_j]}{\sum_{k=1}^{N} \mathbf{W}_{ki} Pr[x \in R_k]} E[x \mid x \in R_j], \end{aligned}$$

which is (4). □

2.2.2 Neighborhood Distortion

Here we incorporate neighborhood distortion measures into WVQ.

Let us define a neighborhood function $\mathcal{N} : Y \times Y \rightarrow \Re$. Typically, $\mathcal{N}(y_i, y_j)$ will yield a value determined by the distance, on the SOFM, between the units corresponding to y_i and y_j. For instance, for the 1-dimensional SOFM analyzed in [13], with a neighborhood radius $r = 1$, $\mathcal{N}(y_i, y_j) = 1$ when $|i - j| \leq 1$, and 0 otherwise.

When the neighborhood function is used to weight a unit's contribution to distortion, the total distortion for an input x is

$$\Delta(x, \mathcal{C}(x)) = \sum_{y \in Y} \mathcal{N}(y_{\mathcal{C}(x)}, y) \times \delta(x, y). \tag{6}$$

The average distortion per codevector is then

$$\begin{aligned} \Delta(x, \mathcal{C}(x)) &= \frac{\sum_{y \in Y} [\mathcal{N}(y_{\mathcal{C}(x)}, y) \delta(x, y)]}{\sum_{y \in Y} \mathcal{N}(y_{\mathcal{C}(x)}, y)} \\ &= \sum_{y \in Y} \left[\frac{\mathcal{N}(y_{\mathcal{C}(x)}, y)}{\sum_{y \in Y} \mathcal{N}(y_{\mathcal{C}(x)}, y)} \times \delta(x, y) \right] \end{aligned} \tag{7}$$

The neighborhood function is incorporated into WVQ by using the neighborhood distortion function to determine the values for the matrix $\mathbf{W}$. For total distortion, we find that

$$\mathbf{W}_{ij} = \mathcal{N}(y_i, y_j). \tag{8}$$

For the average distortion per node, $\mathbf{W}_{ij}$ is the relative (weighted) contribution of y_i in y_j's neighborhood when $\mathcal{Q}(x) = y_j$, i.e.

$$\mathbf{W}_{ij} = \frac{\mathcal{N}(y_i, y_j)}{\sum_k \mathcal{N}(y_k, y_j)}. \tag{9}$$

Note that in (9), $\sum_{i=1}^{N} \mathbf{W}_{ij} = 1$.

The neighborhood distortion functions (6) and (7) can now be expressed using the notation of WVQ, as in (2). We use $\delta(x, y) = |x - y|^2$ so that the WVQ codevector placement condition holds.

For practical purposes, the average meansquare distortion appears to be the more useful measure, because it compensates for the fact that at the boundary of the SOFM fewer nodes contribute to the distortion. Of course, as the size of the SOFM is increased, relative to the size of the neighborhood, such boundary effects are diminished.

2.2.3 A Neighborhood Distortion-Based Learning Algorithm

By incorporating SOFM neighborhoods into WVQ, we obtain the following neighborhood distortion-based algorithm for training SOFM. Unlike traditional SOFM learning, distortion is monotonically reduced, and thus topology preservation is monotonically increased, with this algorithm. Like the GLA, this algorithm produces locally optimal codebooks. When appropriate, simulated annealing and other techniques can be used to produce codebooks closer to the global optimum.

1. Initialize codevectors, set threshold for termination.
2. Loop until decrease in distortion is below threshold.
   (a) Partition the training set using the WVQ partitioning rule.
   (b) Compute new codebook using the WVQ codevector placement condition and the partitioning from step (a)
   (c) Compute the neighborhood distortion summed over the training set, and compute the decrease in distortion compared to the last iteration.

Our current implementation of this algorithm stores the SOFM weight vectors as a linear array, and the $\mathbf{W}$ matrix defines the arrangement of the SOFM units and the neighborhood function. By specifying the weight matrix $\mathbf{W}$, the number of units n, and the number of components in input vector, any SOFM can be trained. This includes SOFMs with units in more than two dimensions.

2.3 Comments

To preserve topology is to preserve the relationships of adjacency and non-adjacency in the input space. Clearly, then, SOFM at best perform a discrete approximation of a topology preserving mapping. But preserving topology is not enough: to be useful, SOFM must also form an accurate representation of the input distribution. Neighborhood distortion functions can be used to train SOFM to achieve these goals to the extent possible. Minimizing neighborhood distortion ensures that units close on the SOFM will have similar weight vectors, which encourages the preservation of adjacency. Likewise, minimizing neighborhood distortion also ensures that units close on the SOFM will not have dissimilar weight vectors, which encourages the preservation of non-adjacency.

We have found that larger neighborhoods tend to support only topology preservation, and smaller neighborhoods tend to support only a good representation of the input distribution. It is well known that training should begin with large neighborhoods and then smaller neighborhoods should be used as training proceeds. Still, the question of how to change the neighborhood distortion function as time proceeds remains an important issue that needs further study.

SOFM's topology preservation capability rests on the assumption that topology will be best preserved when neighborhood distortion is minimized. Under this assumption, topology can be measured by the total distortion over the training set. Another useful measure, which depends only on the SOFM weights, is to sum the neighborhood distortion over the set of weight vectors. If better measures than neighborhood distortion are formulated, perhaps improved learning algorithms will be developed.

Finally, the use of neighborhood distortion measures to train SOFM also yields a criterion for stopping training, and comparing different SOFM, unlike traditional SOFM learning where training is stopped in an ad hoc manner.

3 ASR EXPERIMENTS

All experiments were conducted using the isolated digits subset of the studio quality TI Connected Digit Database [17]. The sampled data was extracted from the 10 Khz sampling rate distribution, and an endpoint detection algorithm [10] was used to determine the endpoints of the utterances. The experiments used utterances from either men, women, or both (men and women). The data from 55 men and 57 women, and another 54 men and 57 women, were used to construct the training sets and testing sets, respectively. Each person uttered two repetitions of each of the 11 digits ("zero" through "nine", plus "oh"). The standard partitioning of data into training and testing sets was used, as given in [17].

In this study we were concerned with exploring the use of the topology preserving properties of SOFM. Therefore we kept the recognition features and system as simple as possible. For example, we did not use dynamic features, and used a simple scheme of linear warping followed by K nearest neighbor classification. For this study we wanted to make the system as sensitive to the SOFM as possible, so that the power of other components of the recognizer did not obscure our comparison of different SOFMs. For practical applications, however, it would be prudent to use, e.g., dynamic time warping, or more informative recodings of the SOFM responses, as in [19][11].

3.1 Feature Generation

Transducer-level features were generated from the sampled data, using a frame length of 200 samples and a shift of 100 samples per frame. The features were a critical-band filterbank; eighth order PLP analysis[9], with and without RPS weighting[7]; zerocrossing rate; and log rms energy. The filterbank was FFT-based, with seventeen digital filters equally spaced on a bark scale from 99 Hz to 5000 Hz [9].

Various square[1] SOFMs were then trained, frame by frame, using the transducer-level features. Inputs were either the seventeen critical band filterbank responses, PLP coefficients, RPS weighted PLP coefficients, or both the zerocrossing rate and log rms energy. Following [3] and [14], each of the the inputs was also normalized to constant length, and the resultant data was used to train additional SOFMs. In all cases the outputs of an SOFM consisted of the Cartesian coordinates of the best matching unit's location on the SOFM.

To test the utility of using line-segment SOFMs to normalize each component of the input vector as an alternative to the constant length input normalization used above, we trained line-segment SOFMs, each on a single component of a transducer-level feature vector. The output of the line-segment SOFM was the position of the best matching unit on the line segment. Then we used the resultant line-segment SOFM responses to construct input patterns for square SOFMs.

Finally, to explore their feature integration capabilities, SOFMs were trained using the outputs of pairs of the previously trained SOFMs.

[1] Herein a square SOFM is two dimensional and has its units arranged in a rectangular tesselation on a unit square, and a line-segment SOFM is one dimensional and has its units arranged on a unit line segment. Since the position of the best matching unit is used for further processing, a square SOFM maps a vector of reals to a finite subset of [0,1]x[0,1], and a line-segment SOFM maps a vector of reals to a finite subset of [0,1]. In this paper, an SOFM is assumed to be square unless otherwise noted.

Representation	Recognition Rate (%)		
	Both	Female	Male
cbfilt	63.9	89.6	91.3
Length—cbfilt	94.8	94.3	95.2
SOFM—cbfilt	95.7	94.1	93.2
PLP	86.4	87.6	87.0
SOFM—PLP	92.3	88.4	93.2
PLP-RPS	88.2	87.9	91.2
SOFM—PLP-RPS	92.6	94.8	93.0
zero_lrms	68.4	69.6	71.2
Length—zero_lrms	72.5	72.7	74.3
SOFM—zero_lrms	73.0	71.5	72.2

Table 1: Recognition rates for SOFMs trained on critical band filterbank responses (cbfilt), PLP, PLP-RPS, or zerocrossing rate and log rms energy (zero_lrms). "Length" indicates that the inputs were normalized to constant length, and "SOFM" indicates that inputs were normalized using line-segment SOFMs, as explained in Section 2.2.

SOFMs used for input		Recognition Rate (%)		
		Both	Female	Male
Length—cbfilt	Length-zero_lrms	93.6	93.4	94.5
Length—cbfilt	SOFM-zero_lrms	94.2	92.7	95.2
SOFM—cbfilt	Length-zero_lrms	96.7	95.4	97.2
SOFM—cbfilt	SOFM-zero_lrms	97.2	97.6	98.2
SOFM—cbfilt	SOFM-PLPRPS	98.3	98.2	98.6

Table 2: Recognition rates for SOFMs trained using pairs of the SOFMs shown in Table 1.

3.2 SOFM Training Strategy

Torkkola and Kokkonen have independently studied the direct use of topology preservation for speech recognition [19]. Their results, when compared to using SOFMs to produce symbols [14], were unfavorable. Upon inspection, however, one finds that even in the final stages of training the neighborhood included all units adjacent to the best matching unit. This was done to maintain good topology preservation. Such SOFM training strategies produce poor codebooks in a vector quantization sense, however, because of the smoothing of codevectors within their respective neighborhoods. So, in [19], topology preservation was presumably obtained, but at the cost of having a codebook which poorly represented the input distribution.

Our viewpoint is that, when using their topology preserving capabilities, the goal of SOFM learning for recognition of studio-quality speech is (ideally) to yield a topology preserving arrangement of an optimal noiseless-channel VQ codebook on the SOFM. To compare these two viewpoints we performed a preliminary experiment, with the male data, using the SOFM normalized critical band filterbank responses as input to another SOFM, and classified patterns using the K nearest neighbor classifier described in Section 3.3. With a neighborhood including the best matching unit and its nearest neighbors, the recognition rate was only 72.9 percent. When the neighborhood including only the best matching unit, the recognition rate increased substantially, to 93.2 percent. Therefore, in subsequent experiments, we used a neighborhood consisting of only the closest unit in the final stages of training.

All square SOFMs had 400 units, arranged with a 20 x 20 rectangular tesselation. All line-segment SOFMs had 101 units. Training always started with neighborhoods including all of the units (to initialize the weights to the centroid of the set of input vectors), and the size of the neighborhoods were slowly decreased until the neighborhood consisted of a single unit.

3.3 Classifier

Since the focus of this paper is a comparative study of various strategies for using SOFMs for ASR, we used a simple K nearest neighbor (KNN) classifier. We thus had to produce fixed length pattern sequences from the SOFM responses over an utterance.

Here we simply linearly expanded or contracted the pattern sequences from each utterance to a constant length of 16 patterns, using linear interpolation of the responses occurring in each of 16 equal intervals of time. The results reported here use $K = 5$, which generally produced the best results.

3.4 Results and Discussion

The results of the experiments are given in Tables 1 and 2. Input normalization clearly improved recognition rates, and line-segment SOFM normalization was as effective as constant length normalization. Furthermore, when SOFM-normalized features were combined (table 2), the recognition rates exceeded 98 percent.

Our use of SOFMs automatically affords a uniform representation for different features. Regardless of the statistics of the input, the outputs of square and line-segment SOFMs have, respectively, identical ranges, and similar variances. This uniform representation proved useful in the above experiments for SOFM based feature integration, and should prove useful for input to other types of neural networks, where input representation is a well-known determinant of network performance [18].

4 CONCLUSIONS

SOFM preserve topology by reducing neighborhood distortion during training. Through the use of appropriate neighborhood distortion functions SOFM can be trained to both preserve topology and to form a good representation of the input distribution. Our SOFM learning algorithm is simple, easy to vectorize, and has a uniform representation for any SOFM; furthermore, unlike traditional SOFM learning, neighborhood distortion is monotonically reduced with each pass through the training set.

There are three main conclusions that can be drawn from the ASR experments:

1. Updating only the closest unit in the final stages of training resulted in a marked improvement in recognition rates.
2. Feature maps can be effectively used to normalize input patterns.
3. Feature maps can be effectively used to integrate features from different sources.

Finally, we note that good performance was achieved with only basic transducer-level features and a rudimentary classification scheme, and so we expect a commensurate increase in performance when we introduce, e.g., dynamic features and more sophisticated classifiers to the system.

References

[1] Bauer, H.-U., and K. R. Pawelzik, "Quantifying the neighborhood preservation of self-organizing feature maps," *IEEE Transactions on Neural Networks*, 3, 570–579, 1992.

[2] De Haan, G. R., "Kohonen Nets for ASR", *STL Symposium*, Osaka, Japan, 1990. Speech Technology Laboratory, 3888 State Street, Santa Barbara, CA 93117 USA.

[3] De Haan, G. R., Eğecioğlu, Ö., and H. J. Wakita, "Improving the performance of backpropagation trained vowel classifiers," *The Journal of the Acoustical Society of America*, 84, supp. 1, paper U4, 1988.

[4] De Haan, G. R., and Ö. Eğecioğlu, "Neighborhood distortion functions and self–organizing feature maps", *International Joint Conference on Neural Networks*, July 1991.

[5] De Haan, G. R., Eğecioğlu, Ö., "Links between self-organizing feature maps and weighted vector quantization," Computer Science Department Technical Report TRCS 92-1, University of California, Santa Barbara, 1992.

[6] Gersho, A., "On the structure of vector quantizers," *IEEE Transactions on Information Theory*, IT-28, 157–166, 1982.

[7] Hanson, B. A., and H. Wakita, "Spectral slope distance measures with linear prediction analysis for word recognition in noise", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35, 968–973, 1987.

[8] Hecht-Nielsen, R., *Neurocomputing*, Addison-Wesley, Reading, MA, 1990.

[9] Hermansky, H., Hanson, B. A., and H. Wakita, "Low-dimensional representation of vowels based on all-pole modeling in the psychophysical domain," *Speech Communication*, 4, 181–187, 1985.

[10] Junqua, J. C., Reaves, B., and B. Mak, "A study of endpoint detection algorithms in adverse conditions: incidence an a DTW and HMM recognizer," *Eurospeech 91*, Vol 3, 1371–1374, 1991.

[11] Kangas, J., Torkkola, K., and M. Kokkonen, "Using SOMS as feature extractors for speech recognition," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Volume II, 341-345, 1992.

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

[13] Kohonen, T., *Self-Organization and Associative Memory*, (Second Edition), Springer-Verlag, Berlin, 1988.

[14] Kohonen, T., "The 'neural' phonetic typewriter," *Computer*, 21, 11–22, 1988.

[15] Kohonen, T., "The self-organizing map," *Proceedings of the IEEE*, 78, 1464–1480, 1990.

[16] Luttrel, S. P., "Codevector density in topographic mappings: scalar case," *IEEE Transactions on Neural Networks*, 2, 427–436, 1991.

[17] NIST CD-ROM Version of the Texas Instruments-developed Studio Quality Speaker-Independent Connected-Digit Corpus.

[18] Pao, Y.-H., *Adaptive Pattern Recognition and Neural Networks*, Addison-Wesley, USA, 1989.

[19] Torkkola, K., and M. Kokkonen,, "Using the topology-preserving properties of SOFMS in speech recognition," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Volume I, 261–264, 1991.

[20] Werbos, P. J., "Generalization of backpropagation with application to a recurrent gas market model," *Neural Networks*, 1, 339–356, 1989.

[21] Zeger, K., *Source and Channel Coding with Vector Quantization*, Ph.D. Dissertation, UC Santa Barbara, 1990.