Feature Maps for Input Normalization and Feature Integration in a Speaker Independent Isolated Digit Recognition System

Gregory R. De Haan[1,2] Ömer Eğecioğlu[2]

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

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

Abstract

We report on the use of the topology preserving properties of Feature Maps for speaker independent isolated digit recognition. The results of recognition experiments indicate that Feature Maps can be effectively used for input normalization, which is an important concern for practical implementations of neural network-based classifiers. Recognition rates can be increased when a third Feature Map is trained to integrate the responses of two Feature Maps, each trained with different transducer-level features. Despite the use of a rudimentary classification scheme, recognition rates exceeded 97 percent for integrated, Feature Map normalized, transducer-level features.

1 Introduction

Feature Maps, as developed by Kohonen [9], have proven to be useful and versatile neural networks, with applications ranging from clustering/classification to control and to optimization. See [11] for a thorough review.

Feature Maps are most often used for clustering, i.e. for producing symbols. Recently, however, there has been an interest in exploring the use of the much-heralded topology preserving capabilities of the Feature Maps for practical applications [1] [16]. The essence of topology preservation is that, after training, the weights of a Feature Map are arranged such that their location on the Map preserves information about the topological relationships in the input space.

In this paper we present the results of speaker independent isolated digit recognition experiments which were designed to explore the use of the topology preservation capability of Feature Maps. First, we propose a training strategy which both preserves topology and forms a good representation of the input space. Second, we explore the use of Feature Maps for input pattern normalization, which is a critical issue when applying neural networks to practical problems. Finally, we explore the use of Feature Maps to integrate information from different sources.

0-7803-0559-0/92 $3.00 © 1992 IEEE

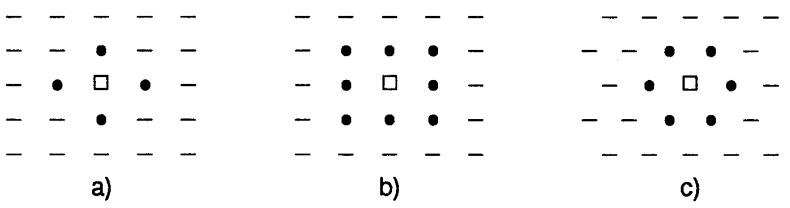

Figure 1: Three cases of nearest neighbor neighborhoods (shown as •) about a given unit (shown as □) in a 2-dimensional Feature Map: a) 4-adjacent rectangular tesselation, b) 8-adjacent rectangular tesselation, c) 6-adjacent hexagonal tesselation. In this paper 8-adjacent rectangular neighborhoods are used.

2 Feature Map Learning

A Feature Map consists of neuron-like units in a bounded d-dimensional space: typically $d = 1$ or $d = 2$. Usually the units are placed uniformly on the map, and, for the two dimensional case, with either a rectangular or hexagonal tessellation. Figure 2 illustrates rectangular and hexagonal tesselations, as well as examples of nearest-neighbor neighborhoods.

For Kohonen learning, the map is trained iteratively, with each unit u_i updating its weight vector y_i. For each iteration, Kohonen's Feature Map training algorithm proceeds as follows:

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

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

with $0 < \alpha_t < 1$. Herein we only consider Euclidean distance. Furthermore, only the neighborhood relationships between the units is used to determine whether a weight is modified or not. These constraints are ubiquitous in the literature (e.g. [1] [9] [10] [11] [16]). In general, other distance measures may be used, and the magnitude of the weight changes can vary as a function of the distance between the units on the Feature Map.

Note that when an input is presented, a given unit u is updated if and only if the closest unit is in u's neighborhood. Thus with Kohonen learning, the expected value of a unit's weight vector is the *centroid* of all input patterns for which either the unit or one of its neighbors wins the competition.

3 Experiments

All experiments were conducted using the isolated digits subset of the studio quality TI Connected Digit Database [12]. The sampled data was extracted from the 10 Khz sampling rate distribution, and a noise-adaptive endpoint detection algorithm described in [8] 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 [13].

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, 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 [7].

Various square[1] Feature Maps were then trained [9], frame by frame, using the transducer-level features. Inputs were either the seventeen critical band filterbank responses, or both the zerocrossing rate and log rms energy. Following [3] and [10], each of the the inputs was also normalized to constant length, and the resultant data was used to train additional Feature Maps. In all cases the outputs of a Feature Map consisted of the Cartesian coordinates of the best matching unit's location on the Feature Map.

To test the utility of using line-segment Feature Maps to normalize each component of the input vector, as opposed to the constant length input normalization used above, we trained line-segment Feature Maps, each on a single component of a (unnormalized) transducer-level feature. The output of the line-segment feature map was the position of the best matching unit on the line segment. Then we used the resultant line-segment Feature Map responses to construct input patterns for square Feature Maps.

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

3.2 Feature Map Training Strategy

Torkkola and Kokkonen have independently studied the direct use of topology preservation for speech recognition [16]. Their results, when compared to using Feature Maps to produce symbols [10], were unfavorable. Upon inspection, however, one finds that Feature Maps were trained with the final stages of training including the nearest neighbors of the best matching unit: the focus was to maintain good topology preservation. Such Feature Map training strategies produce poor codebooks in a vector quantization sense, however, because of the smoothing of codevectors within their respective neighborhoods. So, in [16], topology preservation was presumably obtained[2], but at the cost of having a codebook which poorly represented the input distribution.

Our viewpoint is that, when using the their topology preserving capabilities, the goal of Feature Map learning for recognition of studio-quality speech is (ideally) to arrange an optimal noiseless-channel VQ codebook on the Feature Map so as to preserve topology. In this way the codevectors represent the input space, and topology is preserved. To compare

[1] Herein a square Feature Map is two dimensional and has its units arranged in a rectangular tesselation on a unit square, and a line-segment Feature Map 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 Feature Map maps a vector of reals to a finite subset of [0,1]x[0,1], and a line-segment Feature Map maps a vector of reals to a finite subset of [0,1]. In this paper, a Feature Map is assumed to be square unless otherwise noted.

[2] in [16] there is no mention of a measure to determine how well topology is preserved. See [2] for a discussion of this issue.

Representation	Recognition Rate (%)		
	Both	Female	Male
cbfilt	63.9	89.6	91.3
Length—cbfilt	94.8	94.3	95.2
Feature Map—cbfilt	95.7	94.1	93.2
zero_lrms	68.4	69.6	71.2
Length—zero_lrms	72.5	72.7	74.3
Feature Map—zero_lrms	73.0	71.5	72.2

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

these two viewpoints we performed a preliminary experiment, with the male data, using the Feature Map normalized critical band filterbank responses as input to another Feature Map, 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 Feature Maps had 400 units, arranged with a 20 x 20 rectangular tesselation. All line-segment Feature Maps 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 Feature Maps for ASR, we used a simple K nearest neighbor (KNN) classifier. We thus had to produce fixed length pattern sequences from the Feature Map 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 the 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 Feature Map normalization was as effective as constant length normalization. Furthermore, when Feature Map-normalized features were combined (table 2), the recognition rates were above 97 percent.

Our use of Feature Maps automatically affords a uniform representation for different features. Regardless of the statistics of the input, the outputs of square and line-segment Feature

Feature Maps used for input		Recognition Rate (%)		
		Both	Female	Male
Length—cbfilt	Length-zero_lrms	93.6	93.4	94.5
Length—cbfilt	Feature Map-zero_lrms	94.2	92.7	95.2
Feature Map—cbfilt	Length-zero_lrms	96.7	95.4	97.2
Feature Map—cbfilt	Feature Map-zero_lrms	97.2	97.6	98.2

Table 2: Recognition rates for Feature Maps trained using pairs of the feature maps shown in Table 1

Maps have, respectively, identical ranges, and similar variances. This has been shown useful in the above experiments for Feature Map based feature integration, and should prove to be useful for input to other types of neural networks, where input representation is a well-known determinant of network performance [14].

4 Conclusions

There are three main conclusions that can be drawn from this study:

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.

In addition to further study concerning the main conclusions, we are also studying the following:

1. the use of dynamic (transducer-level) features.
2. running experiments with scalar-input line-segment Feature Maps where the codevectors are equiprobable.
3. the use of more sophisticated classification strategies to deal with the time-varying aspects of speech.

The use of dynamic features [5] is important because such features are ubiquitous in digit recognition[15], and also are useful for recognizing speech in unidealistic conditions (e.g. with added noise, speech produced in noisy conditions), i.e. for practical speech recognizers.

The study of equiprobable codevectors is important because scalar-input Feature Maps are scalar quantizers: when the neighborhood consists of a single unit in the later stages of training, the codevector distribution tends to be proportional to the cube root of the input probability distribution [17], rather than the directly proportional locations of equiprobable codevectors that [6] and [4] claim are ideal.

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 dynamic features and more sophisticated classifiers to the system.

References

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

[2] 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.

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

[4] DeSieno, D., "Adding a conscience to competitive learning," *Proceedings of the 2nd IEEE International Conference on Neural Networks*, Vol. I, 117–124, 1988.S

[5] Furui, S., "Speaker independent isolated word recognition using dynamic features of speech spectrum", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34, 52–59, 1986.

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

[7] 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.

[8] 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.

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

[10] Kohonen, T., "The "neural" phonetic typewriter," *Computer*, 21, 11–22, 1988.

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

[12] Leonard, R. G., "A database for speaker-independent digit recognition," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Volume III, 42.11, 1984.

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

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

[15] Rabiner, L. R., Wilpon, J. g., and F. K. Soong, "High performance connected digit recognition using hidden markov models", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37, 1214–1226, 1989.

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

[17] Zador, P. L., "Asymptotic quantization error of continuous signals and the quantization dimension," *IEEE Transactions on Information Theory*, IT-28, 139–149, 1982.