Self-Organizing Maps of Symbol Strings with Application to Speech Recognition

Teuvo Kohonen and Panu Somervuo

Helsinki University of Technology
Neural Networks Research Centre
Rakentajanaukio 2 C, FIN-02150 Espoo, Finland

Abstract

SOM and LVQ algorithms for symbol strings have been introduced and applied to isolated-word recognition, for the construction of an optimal pronunciation dictionary for a given speech recognizer.

1 Introduction

Speech recognition based on phonemic units produces phoneme strings as output. These are usually not identical with the pronunciation models found in a conventional dictionary. Due to the coarticulation effects, the phones depend on their contexts. The produced phoneme strings also depend strongly on the characteristics of the speech recognizer. The reference strings for a practical pronunciation dictionary used for speech recognition must be optimized so that they classify the produced strings with maximum safety margins.

It has transpired recently that both the SOM and the LVQ algorithms can be designed for the optimal classification of string variables. In the present study, these algorithms were used to construct a pronunciation dictionary for a speech recognition task.

The SOMs are usually defined in metric vector spaces. In SOMs of symbol strings or other nonvectorial representations, the relative locations of the images of the strings on the SOM ought to reflect, e.g., some distance measure, such as the *Levenshtein distance* or *feature distance (FD)* between the strings (for textbook accounts, cf. [1,2]). If one tries to apply the traditional SOM algorithm to such entities, the difficulty immediately encountered is that *incremental learning laws cannot be expressed for symbol strings.* Neither can a string be regarded as a vector.

One of the authors has shown [3] that the SOM philosophy is nonetheless amenable to the construction of ordered similarity diagrams for string variables, if the following ideas are applied:

1. The *Batch Map* principle [2] is used to define learning as *conditional averages over subsets of selected strings.*
2. The "averages" over the strings are computed as *generalized means* or *medians* [4] over the strings.

An advantage, not possessed by the vector-space methods, is further obtained if the feature distance (FD) measure for strings is applied. The best match of the input string against an arbitrary number of reference strings can then be found directly by the so-called *Redundant Hash Addressing (RHA)* method [5,2]. In it, the number of comparison operations, in the first approximation at least, *is almost independent of the number of reference strings.* Construction of very large SOM arrays for strings then becomes possible.

2 Averages of strings

Assume a fundamental set $\mathcal{S}$ of any kind of items $x(i)$ and let $d[x(i), x(j)]$ be some distance measure between $x(i), x(j) \in \mathcal{S}$. The *(generalized) mean* m over $\mathcal{S}$ is defined as the item that satisfies the condition

$$\sum_{x(i)\in\mathcal{S}} d^2[x(i), m] = \min! \tag{1}$$

On the other hand, the *(generalized) median* M over $\mathcal{S}$ shall satisfy the condition

$$\sum_{x(i)\in\mathcal{S}} d[x(i), M] = \min! \tag{2}$$

Thereby m and M need not belong to $\mathcal{S}$. If m and M are restricted to the elements of $\mathcal{S}$, then they may be called the *set mean* and *set median*, respectively. It is relatively easy to show that the usual (set) median of real numbers is defined by (2), if $d[x(i), x(j)] = |x(i) - x(j)|$.

The set mean and the set median are found easily, by computing all the mutual distances between the given elements, and searching for the element that has the minimum sum of squares of the distances, or the minimum sum of distances, respectively, from the other elements. The (generalized) mean and median are then found by systematically varying each of the symbol positions of the set mean or median, making artificial string errors of all the basic types (replacement, insertion and deletion of a symbol), over the whole alphabet, and accepting the variation if the sum of squares of the distances or the sum of distances from the other elements is decreased. The computing time is usually quite modest; even with the 50 per cent error rate reported here, the mean and median can be found in the immediate vicinity of the set mean and set median, respectively, in one or a few cycles of variation.

Table 1 gives two typical examples of sets of erroneous strings generated by a random-number-controlled choice of the errors. The set mean, set median, mean, and median have then been computed, using the Levenshtein distance and the bigram feature distance method, respectively.

3 Self-Organizing Maps of symbol strings

In order to construct a SOM for strings, a number of additional problems have to be solved. One of them is *initialization* of the SOM with proper strings. It would be best if the initial values were already roughly ordered. Such partly ordered initial values of the strings can be picked up from the *Sammon projection* [2,6] of a sufficient number of representative input samples. Fig. 1 in Sec. 4 exemplifies this kind of initialization.

The conventional Batch Map computing steps [2] are applicable to string variables almost as such:

1. For the initial reference strings, take, for instance, samples that are ordered two-dimensionally in the Sammon projection.
2. For each map unit i, collect a list of copies of all those sample strings whose nearest string belongs to the topological neighborhood set N_i of unit i.
3. Take for each new reference string the mean or median over the respective list.
4. Repeat from 2 a sufficient number of times, until the reference strings are no longer changed in further iterations.

Notice that steps 2 and 3 need less memory if at step 2 we only make lists of the training samples x at those units that have been selected for winner, and at step 3 we form the mean *over the union of the lists* that belong to the neighborhood set N_i of unit i.

Table 1: Means and medians of garbled strings. *LD*: Levenshtein distance; *FD*: feature distance. The errors of all different types occured with equal probabilities in the garbled strings.

Correct string: **MEAN**
Garbled versions (50 per cent errors):

1. MAN	6. EN
2. QPAPK	7. MEHTAN
3. TMEAN	8. MEAN
4. MFBJN	9. ZUAN
5. EOMAN	10. MEAN

Set mean *(LD)*:	**MEAN**	**Set median *(LD)*:**	**MEAN**
Mean *(LD)*:	**MEAN**	**Median *(LD)*:**	**MEAN**
Set mean *(FD)*:	**MEAN**	**Set median *(FD)*:**	**MEAN**
Mean *(FD)*:	**MEAN**	**Median *(FD)*:**	**MEAN**

Correct string: **HELSINKI**
Garbled versions (50 per cent errors):

1. HLSQPKPK	6. HOELSVVKIG
2. THELSIFBJI	7. HELSSINI
3. EOMLSNI	8. DHELSIRIWKJII
4. HEHTLSINKI	9. QHSELINI
5. ZULSINKI	10. EVSDNFCKVM

Set mean *(LD)*:	HELSSINI	Set median *(LD)*:	HELSSINI
Mean *(LD)*:	**HELSINKI**	**Median *(LD)*:**	**HELSINKI**
Set mean *(FD)*:	HELSSINI	Set median *(FD)*:	HELSSINI
Mean *(FD)*:	HELSSINI	Median *(FD)*:	HELSSINI

The problems in initializing the SOM, as well as the reasonably high computing load in evaluating the string distances and various averages have caused that the most advantageous learning strategy in this method is to start with a very small SOM, and after its preliminary convergence, to *halve* the grid spacings intermittently by introducing new nodes in the middle of the old ones. Training is then continued. A new problem then arises: the initial values for the middle nodes should be computed as averages of the old ones, but how can we interpolate between strings? The solution is that if we input all the available samples to the smaller SOM which has already converged, and construct the lists of input strings at the matching nodes like in a Batch Map, then for the intermediate value to be used for the initialization of each middle node one can take the average (mean or median) over the union of the lists collected for the neighboring nodes. After the first "expansion" and initialization of the middle nodes, the larger SOM is again taught by the available samples and "expanded," the new middle nodes are initialized in the same way, and so on, until the wanted size of the SOM is achieved.

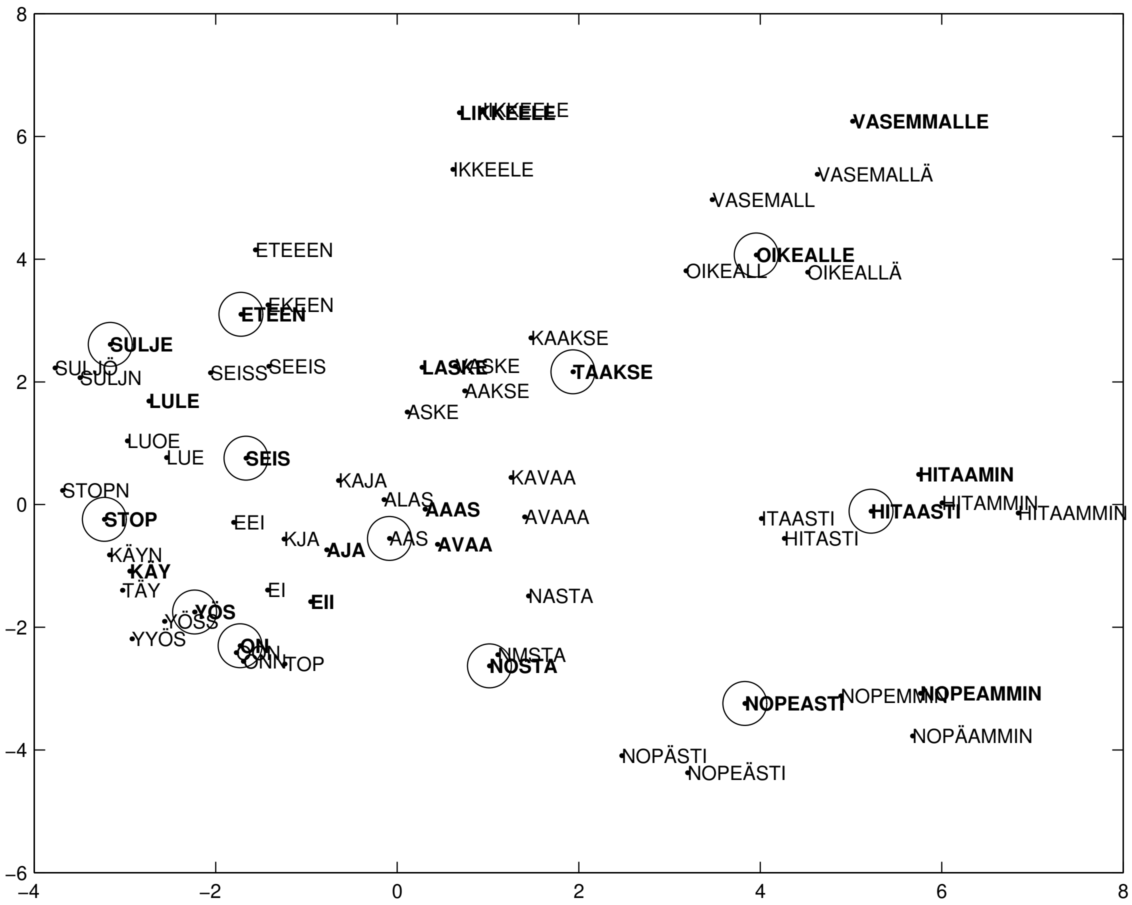

Figure 1: Sammon projection of phoneme strings. The set means of the classes have been printed in boldface. Three string samples in addition to the mean string have also been shown in the picture. The twelve mean strings that were used as initial strings in the first SOM experiment have been encircled.

4 String SOMs for phonemic transcriptions

We have made SOMs of strings for natural phonemic transcriptions produced by the speech recognition system similar to that reported in [7]. As feature vectors we used concatenations of three 10-dimensional mel-cepstrum vectors computed at successive intervals of time 50 ms in length. The phoneme-recognition and phoneme-decoding part was first tuned by the speech of 9 male speakers using a 350-word vocabulary, after which the parameters of the system were fixed. The phoneme strings used in the following experiment were then collected from 20 speakers (15 male speakers and 5 female speakers). The string classes represented 22 Finnish machine command words shown in Table 2. Finnish is pronounced almost like Latin.

The Sammon projection of the pronunciations, from which the initial values for a 3 by 4 SOM were picked, is shown in Fig. 1. After that the map was enlarged to the size of 7 by 5 units, the initial values to the interstitial units were interpolated as told above, and the map was trained. After that the map was expanded to the final size of 13 by 9 units and trained again. The final map is shown in Fig. 2.

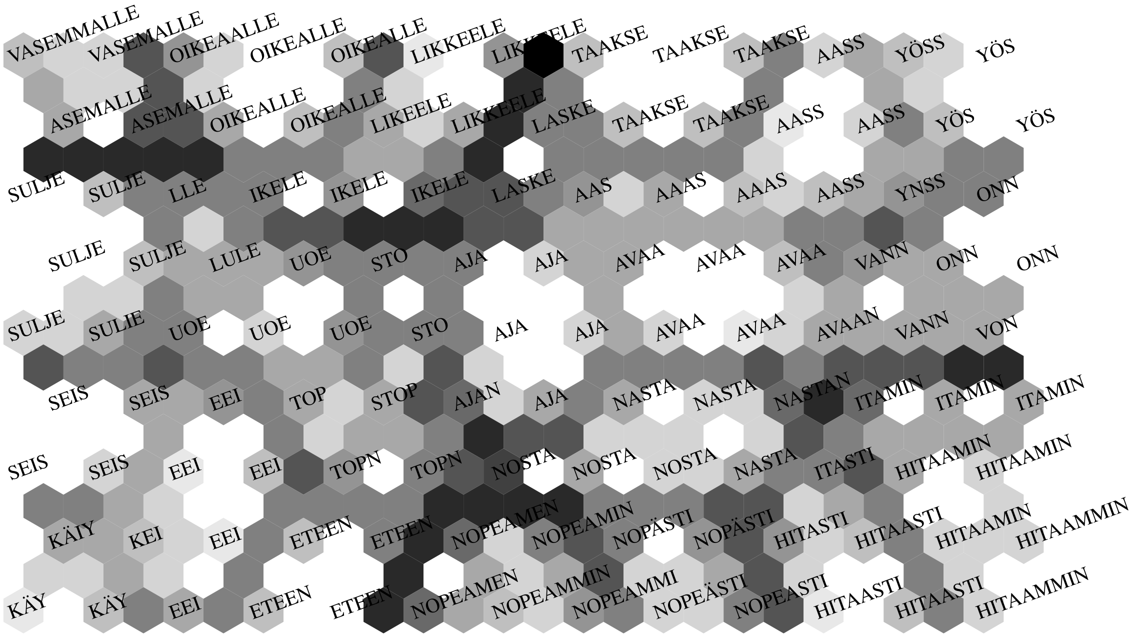

Figure 2: A 13 by 9 unit string-mean SOM. The shades of gray represent distances between neighboring reference vectors; dark means large distance, white small distance, respectively.

Table 2: List of the Finnish commands used in the experiment.

KÄY	(run)
AJA	(drive)
SEIS	(stop)
STOP	(stop)
YLÖS	(up)
ALAS	(down)
VASEMMALLE	(left)
OIKEALLE	(right)
ETEEN	(forward)
TAAKSE	(backward)
LIIKEELLE	(move)
HITAASTI	(slowly)
HITAAMMIN	(slower)
NOPEASTI	(fast)
NOPEAMMIN	(faster)
NOSTA	(lift)
LASKE	(let down)
AVAA	(open)
SULJE	(close)
ON	(yes)
EI	(no)
LUE	(read)

Table 3: Word recognition experiments with 20 speakers. Error percentages.

Mean-SOM

training set	test set
3.6	5.1

Median-SOM

training set	test set
3.0	4.4

5 Multi-speaker word recognition experiments

The *multi-speaker word recognition* experiments for the 20 speakers were carried out using smaller (9 by 9) hexagonal SOM lattices than in the previous examples. The string averages at its nodes were used as a pronunciation dictionary. The training and test sets in the real word recognition experiment consisted of 880 statistically independent transcriptions each. The recognition results are given in Table 3.

The classification accuracy of a usual SOM can be improved by supervised learning, fine tuning the reference vectors by *Learning Vector Quantization (LVQ)* (cf., e.g., [2]). It can be shown that a particular kind of LVQ can be constructed for strings, too [3]. The error percentages for the test set after LVQ fine tuning of the SOMs were reduced to 3.4 per cent for the mean-string SOM, and to 3.8 per cent for the median-string SOM, respectively.

For comparison, if the conventional (linguistic) phonemic transcriptions were used as reference strings, the error percentage would have remained higher: 6.1 per cent for the test set. So the accuracy with LVQ training is almost two times better.

References

[1] T. Kohonen. *Self-Organization and Associative Memory.* Springer Series in Information Sciences, vol. 8, Springer, Heidelberg, 1984.

[2] T. Kohonen. *Self-Organizing Maps*, Springer Series in Information Sciences, vol. 30, Springer, Heidelberg, 1995.

[3] T. Kohonen. Self-organizing maps of symbol strings. Report A42, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland, 1996.

[4] T. Kohonen. Median strings. *Patt. Rec. Lett.*, 3:309-313, 1985.

[5] T. Kohonen. *Content-Addressable Memories.* Springer Series in Information Sciences, vol. 1, Springer, Heidelberg, 1980.

[6] T. Kohonen, J. Hynninen, J. Kangas, and J. Laaksonen. SOM_PAK: The self-organizing map program package. Report A31, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland, 1996.

[7] K. Torkkola, J. Kangas, P. Utela, S. Kaski, M. Kokkonen, M. Kurimo, and T. Kohonen. Status report of the Finnish phonetic typewriter project. In *Artificial Neural Networks*, T. Kohonen et al. (eds.), Elsevier, Amsterdam, vol. 1, pp. 771-776, 1991.