5 Self-Organizing Maps of Symbol Strings

Teuvo Kohonen and Panu Somervuo

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

It has recently transpired [3] that the SOM philosophy is amenable to the construction of ordered similarity diagrams for string variables, too. This method applies the following idea, earlier partly reported in Sec. 4: The *Batch Map* principle [2] (cf. also Sec. 1) is used to define learning as recursively computed *set medians, generalized medians, set means*, or *generalized means* [4] over sets of strings.

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

Let us recall that the *set median* M over $\mathcal{S} = \{X(t)\}$ was defined by

$$\sum_t d[X(t), M] = \min! \tag{54}$$

where $d[X(i), X(j)]$ is the general distance between elements $X(i), X(j) \in \mathcal{S}$, and $M \in \mathcal{S}$. Similarly, the *set mean* m over $\mathcal{S}$ shall satisfy the condition

$$\sum_t d^2[X(t), m] = \min! \tag{55}$$

The *generalized median* and the *generalized mean* are then defined to result from the conditions (54) and (55) when M and m are not restricted to belong to $\mathcal{S}$. If $X(t) \in R^n$, and if m need not belong to $\mathcal{S}$, m is simply the arithmetic mean of the $X(t)$.

The basic types of error that may occur in strings of discrete symbols are: (1) replacement, (2) insertion, (3) deletion of a symbol. (Interchange of two consecutive symbols can be reduced to two of these operations.) An insertion or deletion error changes the relative position of all symbols to the right of it, whereupon, e.g., the most trivial distance between strings of symbols, the Hamming distance is not applicable. There are at least two categories of distance measures that take into account the "warping" of strings: (1) *Levenshtein distance*, which usually computes the minimum number of editing operations (replacements, insertions, and deletions of symbols) needed to change one string into another; these operations can also be weighted in many ways; (2) comparison of strings by their *local features*, e.g., substrings of N consecutive symbols (N-grams), whereupon the respective local features are said to match only if their relative position in the two strings differs in no more than a prespecified number of positions. The string lengths can also be taken into account [1, 2].

The set median and the set mean for strings are found easily, by computing all the mutual distances between the given strings, and searching for the string that has the minimum sum of the distances, or the minimum sum of squares of the distances, respectively, from the other strings. The generalized median and the generalized mean are then found by systematically varying each of the symbol positions of the set median or the set mean, making 'errors' of all the three types over the whole alphabet, and checking whether the sum of the distances or the sum of squares of the distances from the other elements is decreased. The computing time is usually quite modest; even with the 50 per cent error rate discussed here, the generalized median and the generalized mean can be found in the immediate vicinity of the set median and the set mean, respectively, in one or a couple of cycles of variation.

A number of additional problems has to be solved, too. One of them is *initialization* of the SOM with proper strings.

It is possible to initialize a usual vector-space SOM by random vectorial values. We have also been able to obtain organized SOMs for string variables, starting with random reference strings. However, it is of a great advantage if the initial values are already ordered, even roughly, along with the SOM array.

Ordered, although not yet optimal initial values of the strings can be picked up from the *Sammon projection* [2,6] of a sufficient number of representative input samples. Another partial problem is *interpolation* between strings, especially if the dimensions of the SOM are changed during learning, as made in this work.

The most advantageous learning strategy for 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. If we input all the available samples to the smaller SOM and construct the partial lists at the matching nodes, then for the intermediate value to be used for the initialization of each middle node, we can take the average (median or mean) 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.

SOMs of strings have been made for 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 nine 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 five female speakers). The string classes represented 22 Finnish command words. Finnish is pronounced almost like Latin. The results are shown in Fig. 2.

The classification accuracy of a usual SOM can be improved by supervised learning, fine tuning the reference vectors by the *Learning Vector Quantization (LVQ)* (cf.,

Table 2: Medians and means of garbled strings. *LD*: Levenshtein distance; *FD*: feature distance

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 median *(LD)*:	**MEAN**	**Set mean *(LD)*:**	**MEAN**
Generalized median *(LD)*:	**MEAN**	**Generalized mean *(LD)*:**	**MEAN**
Set median *(FD)*:	**MEAN**	**Set mean *(FD)*:**	**MEAN**
Generalized median *(FD)*:	**MEAN**	**Generalized mean *(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 median *(LD)*:	HELSSINI	Set mean *(LD)*:	HELSSINI
Generalized median *(LD)*:	**HELSINKI**	**Generalized mean *(LD)***:	**HELSINKI**
Set median *(FD)*:	HELSSINI	Set mean *(FD)*:	HELSSINI
Generalized median *(FD)*:	HELSSINI	Generalized mean *(FD)*:	HELSSINI

e.g., [2]). It can be shown that a particular kind of LVQ is able to fine tune strings, too.
In accordance with the Batch-LVQ1 procedure introduced in Ref. [3] and also expounded in the first article of this report, we obtain the Batch-LVQ1 for strings by application of the following computational steps:

1. For the initial reference strings take, for instance, those strings obtained in the preceding SOM process.

2. Input the classified sample strings once again, listing the strings as well as their class labels under the winner nodes.

3. Determine the labels of the nodes according to the majorities of the class labels in these lists.

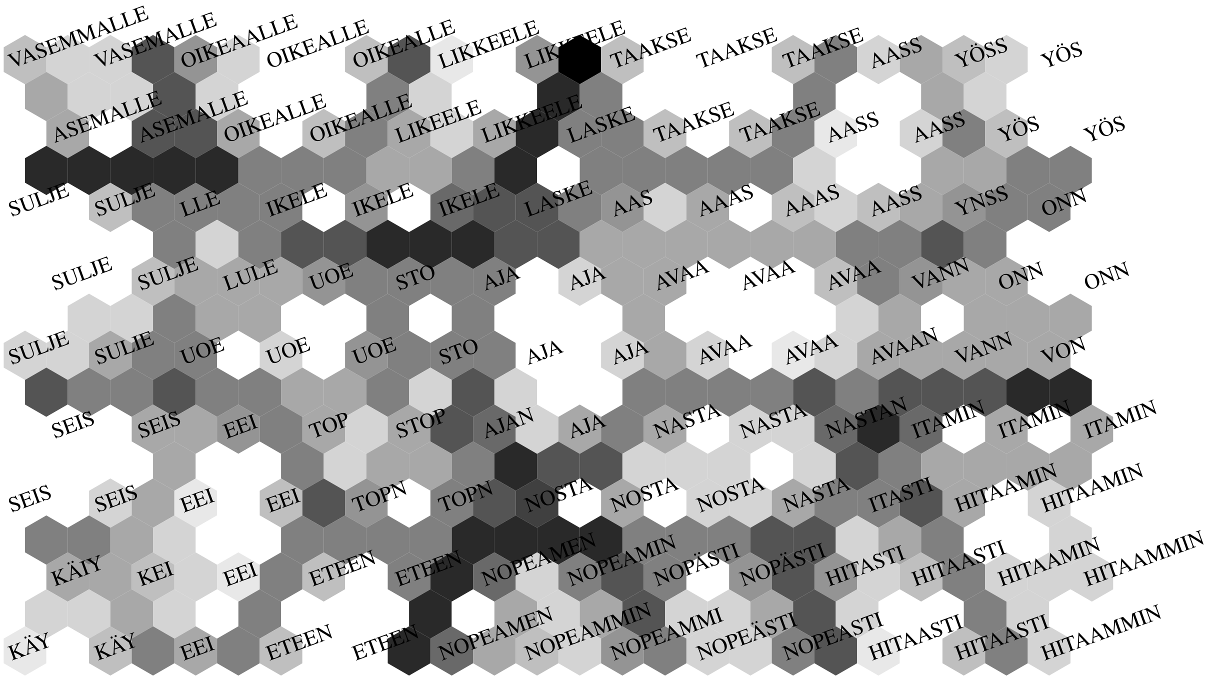

Figure 2: A 13 by 9 unit string-mean SOM. The shades of gray represent distances between neighboring reference vectors; dark means large distance, white small distance, respectively.

4. For each string in these lists, provide its distance (or its square of the distance) from every other string in the same list with the plus sign, if the class label of the latter sample string agrees with the label of the node, but with the minus sign if the labels disagree.

5. Take for the new value of the reference string the string that has the smallest sum of expressions defined at step 4 with respect to all the other strings in the respective list. Continue by systematically varying each of the symbol positions by replacement, insertion, and deletion of a symbol accepting the variation if the sum of expressions defined at step 4 is decreased. Take the best variation for the new reference string.

6. Repeat steps 1 through 5 a sufficient number of times.

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. After training of the SOMs, seven rounds of fine tuning by LVQ1 were performed. The training and test sets consisted of 880 words each. The recognition results are given in Table 3.

We can see from the experiments that the SOM alone may already yield a reasonably high recognition accuracy. For comparison, if the correct (linguistic) phonemic transcriptions had been used as reference strings, the error percentage would have remained higher: 5.8 per cent.

Table 3: Recognition experiments. Average error percentages of four independent runs

Median strings

	training set	test set
SOM only, generalized median	4.3	4.5
SOM only, set median	3.7	3.7
SOM + LVQ1	3.2	3.3

Mean strings

	training set	test set
SOM only, generalized mean	4.1	4.4
SOM only, set mean	4.0	4.0
SOM + LVQ1	2.6	2.7

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.