ELSEVIER

Neurocomputing 21 (1998) 19–30

NEUROCOMPUTING

Self-organizing maps of symbol strings

Teuvo Kohonen*, Panu Somervuo

Helsinki University of Technology, Neural Networks Research Centre, P.O. Box 2200, FIN-02015 HUT, Finland

Accepted 26 May 1998

Abstract

Unsupervised self-organizing maps (SOMs), as well as supervised learning by Learning Vector Quantization (LVQ) can be defined for string variables, too. Their computing becomes possible when the SOM and the LVQ algorithms are expressed as batch versions, and when the average over a list of symbol strings is defined to be the string that has the smallest sum of generalized distance functions from all the other strings. © 1998 Elsevier Science B.V. All rights reserved.

Keywords: Self-organizing map; Learning vector quantization; String clustering

1. Introduction

The self-organizing maps (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, whereby 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* (LD) or *feature distance* (FD) between the strings (for textbook accounts, cf. [2,4]). 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. One of the authors has shown [5] 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 [4] is used to define learning as a succession of *conditional averages over subsets of selected strings.*

* Corresponding author.

PII S0925-2312(98)00031-9

2. The "averages" over the strings are computed as generalized *means* or *medians* [3] over the strings.

An advantage, not possessed by the vector-space methods, is obtained if the feature distance (FD) measure for strings is applied. The best match between the input string against an arbitrary number of reference strings can be found directly by the so-called *redundant hash addressing* (RHA) method [7,1,4]. 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

The SOM is a *similarity diagram of complex entities*. Many different kinds of entities are amenable to SOM representation. In the first place one may think of ordered sets of numbers that stand for sets of signals, measurements, or statistical indicators. It can be shown, however, that the entity to be ordered can be much more general. If x and y are any entities, a sufficient condition for them to be mapped into a SOM diagram is that some kind of symmetric *distance function* $d = d(x,y)$ is definable for all pairs (x,y).

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

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

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

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

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

The basic types of error that 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 the above basic operations in many ways.) An insertion or deletion error changes the relative position of all symbols to the right of it, whereby, e.g., the Hamming distance is not applicable. There are at least two categories of distance measures that take into account the "warping" of strings: (1) *dynamic programming*, 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 be weighted in many ways; (2) comparison of strings by their *local features*, e.g., substrings of N consecutive symbols (N-grams), whereby the respective local features are said to match only if their relative position in the two

strings differs in no more than, say, p positions; the distance of the strings is then a function of their lengths and the matching score.

The set mean and the set median are then 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; the mean and median can be found in one or a few cycles of variation.

2.1. Levenshtein distances

The most familiar distance between the strings may be their *Levenshtein distance* (LD), which for strings A and B is defined as

$$\mathrm{LD}(A,B) = \min\{a(i) + b(i) + c(i)\}. \tag{3}$$

Here B is obtained from A by $a(i)$ replacements, $b(i)$ insertions, and $c(i)$ deletions of a symbol. There exists an infinite number of combinations $\{a(i), b(i), c(i)\}$ to do this, and the minimum can be sought, e.g., by a dynamic-programming method.

Since the various types of error occur with different probabilities and depend on the occurring symbols, the comparison operation is usually more reliable if, in the distance measure, the editing operations are provided with different weights. This then leads to the *weighted Levenshtein distance* (WLD) [9]

$$\mathrm{WLD}(A,B) = \min\{pa(i) + qb(i) + rc(i)\}, \tag{4}$$

where the coefficients p, q, and r may be obtained from the so-called *confusion matrix* of the alphabet, as the inverse probabilities for particular types of error to occur.

2.2. Feature distance

An N-gram is a substring of N consecutive symbols; usually $N = 2$ (bigram or digram), or $N = 3$ (trigram). It is a matter of choice whether the leading and/or trailing blanks are included in the original string, or whether some of the N-grams are defined end-around with respect to the original string. The "window" from which the N-gram is picked up shall move over the original string so that the N-grams overlap partly.

Let n_a, n_b be the numbers of N-grams in strings A and B, respectively, and let e be the number of matching N-grams. The *feature distance* FD [7] is defined as

$$\mathrm{FD}(A,B) = \max(n_a,n_b) - e. \tag{5}$$

This measure has been studied extensively in the context of speech recognition where it has compared favorably with the WLD. While it may sometimes yield a slightly

inferior classification accuracy with respect to the latter, it has one significant merit: There exists an extremely fast searching and comparing method based on the FD. This method, called the *redundant hash addressing* (RHA) [7,8], locates the best-matching candidates directly without scanning all the reference strings. The speed of comparison can be orders of magnitude higher than when conventional methods are used.

2.3. Examples of means and medians of garbled strings

Table 1 gives two typical examples of sets of erroneous strings produced by a random-number-controlled choice of the errors. The probabilities for each type of error to occur were thereby equal. The set mean, set median, mean, and median have been computed using the Levenshtein distance and the bigram feature distance method, respectively.

Table 1
Means and medians 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 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

3. Computation of SOMs for strings

3.1. Initialization of the SOM for strings

It is possible to initialize a usual vector-space SOM by random vectorial values. We have been able to obtain organized SOMs for string variables, too, starting with random reference strings. However, it is of a great advantage if the initial values are already ordered, even roughly, along the SOM array. Then one can use a smaller, even fixed-size neighborhood set. In the usual SOM, for best results, one can select for the initial values, say, a regular two-dimensional ordered sequence of values picked up from a linear subspace spanned by the two largest components in the Karhunen–Loève expansion [6]. The strings, however, are discrete variables, and the Karhunen–Loève expansion for them does not exist. Now we can make use of the fact that the set mean or set median is one of the input samples. It will be possible to define ordered strings for the initial values, if one first forms the *Sammon projection* [11]; for textbook accounts, cf. Refs. [4,6]. From the projection of a sufficient number of representative input samples it is then possible to pick up manually a subset of samples that seem to be ordered two dimensionally.

3.2. The batch map for strings

The conventional batch map computing steps [4] of the SOM 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 those sample strings to whom the reference string of unit i is the nearest reference string.
3. For each map unit i, take for the new reference string the mean or median over the union of the lists that belong to the topological neighborhood set N_i of unit i.
4. Repeat from 2 a sufficient number of times, until the reference strings are no longer changed in further iterations.

Comment. It may sometimes occur that the corrections lead to a "limit cycle" whereby the reference strings oscillate between two alternatives, the latter usually forming a tie in winner search. In such a case the algorithm must be terminated somehow.

3.3. Maps of phoneme strings

The first stages of a speech recognition system are often made to produce 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 pronunciation dictionary

used for speech recognition must be optimized so that they classify the produced strings with maximum safety margins.

We have made SOMs of strings for phonemic transcriptions produced by the speech recognition system similar to that reported in Ref. [12]. 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 turned by the speech of nine male speakers using a 350-word vocabulary, after which the parameters of the system were fixed. 1760 phoneme strings used in the following experiment were collected from 20 speakers (15 male speakers and 5 female speakers). The string classes represented 22 Finnish command words. Finnish is pronounced almost like Latin. The speech recognizer was a vocabulary-free phoneme recognizer that had no grammatical restrictions to its output.

4. Batch-LVQ

The supervised-learning algorithms called the *learning vector quantization* (LVQ) can be used to fine tune the SOM reference vectors for best class separation. In the basic LVQ, only the winner node, but not its topological neighbors, are updated. (Thereby, however, some degree of the topological ordering may be lost; in order to sustain the ordering, the LVQ learning steps can be applied to the winner *and* its neighbors, as in the SOM.) In accordance with the definitions in Section 2 we may now express the "mean" m^* over $\mathscr{S}$ as

$$\sum_{x(i)\in\mathscr{S}} s(i)\, d^2[x(i), m^*] = \min!, \tag{6}$$

where $s(i) = +1$ if $x(i)$ and m belong to the same class, but $s(i) = -1$ if $x(i)$ and m belong to different classes. The corresponding "median" M^* is then defined by

$$\sum_{x(i)\in\mathscr{S}} s(i)\, d[x(i), M^*] = \min!. \tag{7}$$

4.1. Batch-LVQ1 for strings

In accordance with the Batch-LVQ1 procedure introduced in Ref. [5] and also expounded in the first article of this issue, 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.

4. For each string in these lists, compute an expression equal to the left side of Eq. (6) or (7), respectively, where the square of the distance (or the distance itself) of the string from every other string in the same list is provided 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 set mean (set median) in each list the string that has the smallest sum of expressions defined at step 4 with respect to all the other strings in the respective list. Compute the mean (median) by systematically varying each of the symbol positions in the set mean (set median) by replacement, insertion, and deletion of a symbol, accepting the variation if the sum of distances (provided with the same plus and minus signs that were used at the previous step) between the new mean (median) and the sample strings in the list is decreased. Take the corresponding mean (median) for the new reference string.
6. Repeat (steps 1 through 5) a sufficient number of times.

5. Recognition experiments

Multi-speaker word recognition experiments for 20 speakers were carried out using 9×9 unit SOM lattices. For some of the experiments, the SOM reference vectors were fine tuned by the Batch-LVQ1 principle described above. The reference strings of the SOM were used as a pronunciation dictionary for the given speech recognition task. Phoneme strings were divided into four sets, each of them containing 440 strings. Two of the sets were used for training, one set was used for training validation and the remaining fourth set was used as an independent test set. The weighted Levenshtein distance was used, and its weights were computed from the confusion matrix of the training set. The recognition experiments were repeated four times, each time having different combinations of the sets for training and testing.

5.1. Tie-break rules

Notice that all strings are discrete-valued entities. Therefore, ties in comparison may occur frequently, and the convergence to the final values in learning may not be particularly fast if the map is big. Ties may occur (1) in the winner search, (2) in finding the (set) mean or median, and (3) when giving a label to a node.

In the winner search, two different kinds of ties may occur. If the string classes over which the set mean or set median has to be formed are disjoint, the set mean or set median string of their union may be equal to the set mean or set median of one of the classes. In learning, this may result in copying the same string into all units of the union of these classes, especially if there exist duplicates of these strings in the training set. Another type of tie may occur if the strings are very short. Then the distances between the strings may be equal even if they represent different classes. However, if the weighted Levenshtein distance is used, the number of this kind of tie remains

small. While the latter kind of tie is more related to the feature and distance function selection, solving the former kind of tie is more fundamental. In the present work we had initially ordered strings, but if the strings had been unordered, and if a large neighborhood is used, the amount of identical reference strings may be considerable, and choosing the best-matching unit among the equidistant items at random is not advisable, because this would hinder the ordering and convergence of the map. The most advantageous strategy is then to use the topological neighborhood of the equidistant items for solving the tie. The new distance to be compared is computed as a sum of distances from the sample string to all the reference strings in the neighborhood of the equidistant item divided by the number of nodes in the neighborhood (the border items of the map are thus handled equally compared to the items in the middle of the map). If ties still exist, a larger neighborhood is used, and this is continued until there is only one winner or the whole map has been used as a neighborhood. In the experiments, the winner search ties vanished entirely by using this method. The tendency of forming large areas with identical reference strings is also reduced, if duplicate strings are removed from the training set; this, however, then changes the statistics of the data. The number of identical reference strings is typically very small if the true means and medians are used instead of the set means and set medians. This is because the generalized means and medians are able to interpolate between disjoint string classes.

If a tie occurs in the set mean or set median computation, one might speculate that the final average string might be selected according to the length of the string. In our experiments, however, there were no benefits for favoring either shorter or longer strings.

In labeling, majority voting may also result in a tie. One can then select the label according to its length or alternatively select it from the ties at random. In our experiments random selection gave slightly better results compared to favoring either short or long labels. One possibility to solve the ties in labeling is to remove all those units from the map.

It may be worth noting that solving all ties does not necessarily mean good recognition accuracy. However, a consistent way to solve the ties is beneficial for the convergence of the map.

5.2. Recognition accuracies before and after supervised fine tuning by the Batch-LVQ

In order to guarantee a good statistical accuracy in recognition, the number of data strings in the training set must be sufficiently large compared to the map size. Above, the SOMs used for the recognition experiments had 81, i.e., 9×9, nodes corresponding to about 11 training samples per node, which is just about sufficient.

The results given in Table 2 are both for means and medians, and for set means and set medians, respectively. For the sake of robustness, if a tie occurred in the winner search, in both SOM and LVQ training the string was added to the lists of all equidistant units. This was justified because the initial map was already ordered and all string classes were represented on it. The map was initialized by 81 strings picked up from the Sammon projection of 110 strings in the training set (five strings from

Table 2
Recognition experiments. Average error percentages of four independent runs

	Training set	Test set
Mean strings		
SOM only, mean	4.1	4.4
SOM only, set mean	4.0	4.0
SOM + set LVQ1	3.5	3.8
SOM + LVQ1	2.6	2.7
Median strings		
SOM only, median	4.3	4.5
SOM only, set median	3.7	3.7
SOM + set LVQ1	3.4	3.5
SOM + LVQ1	3.2	3.3

each class). The neighborhood during SOM training was fixed, containing only the four nearest map units to each map unit. It was possible to use such a narrow (fixed-size) neighborhood, because, as said, the map was initially ordered by the Sammon projection.

Although the expression in Eq. (1) or (2) is minimum for the true (generalized) mean and median, respectively, after unsupervised learning the simpler set mean and set median produced better recognition results. The explanation for this is that in the unsupervised learning the (generalized) mean and median smoothen the border between two neighboring string classes on the map. In accordance with this, the number of ties in the labeling was considerably higher for the (generalized) means and medians than for the set means and set medians. In the labeling phase, a tie in the majority voting was solved by taking the label at random from ties. In classification, a tie occurred if no majority class among the equidistant units was found. Then the winner was chosen randomly among the equidistant units. In the experiments where the SOM was trained using the means, the number of ties in the classification was particularly high: they occurred in 0.9 per cent of the strings to be classified on the average. But because the learning was unsupervised at this stage, no corrections to the reference strings were made. In other experiments the number of classification ties was negligible, even zero.

After training of the SOMs using the means and medians, LVQ fine tuning was applied. Now the true (generalized) mean and median performed better than their simpler counterparts set mean and set median, respectively. Fig. 1 shows an example of the training for the mean strings.

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.

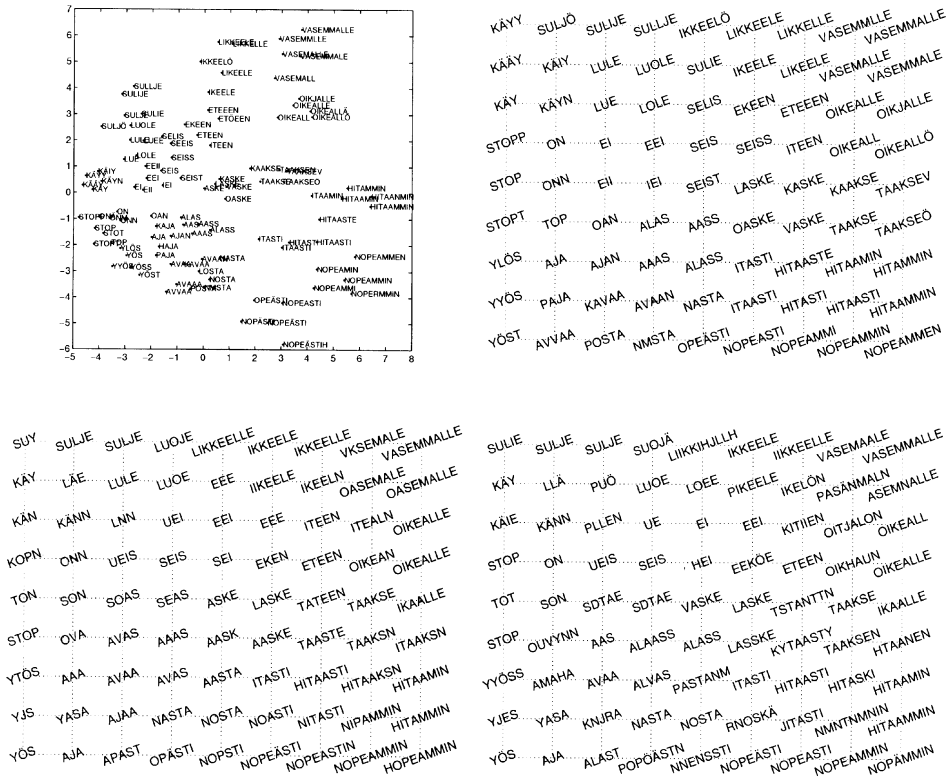


Fig. 1. (Top left) Sammon projection of phoneme strings, (top right) initialization of the SOM for the mean strings, (bottom left) the reference strings of the SOM after 10 batch rounds, (bottom right) the reference strings after five batch rounds of LVQ1 fine tuning.

6. Discussion

If reference strings are used for classification, one must check that all string classes are represented on the SOM. Eventual missing classes can be enforced to the SOM, e.g., by repeating the corresponding samples in the training set, although this might change the input statistics on their part. We can see from the experiments that the SOM alone may already yield a reasonably high recognition accuracy, whereas the improvements due to LVQ1 are not as consequent as with vectorial patterns. Neither were there big differences between the mean and median experiments.

The experimented construction of an optimal pronunciation dictionary for the given speech recognition task was carried out as a demonstration of the algorithms. It is interesting to find out that some self-organizing algorithms are applicable to genuine string variables, too. The self-organizing map for symbol strings allows clustering and partitioning the string space according to the statistics of the input strings. This gives thus solution to the "String Taxonomy Problem" [10]. In addition

to the clustering properties, the averaging in the SOM algorithm removes "noise" from the strings and erroneous strings can be processed. Clustering and classification of general sequential nonvectorial patterns is possible and potential applications can be found, e.g. in molecular biology, telecommunications, and speech recognition.

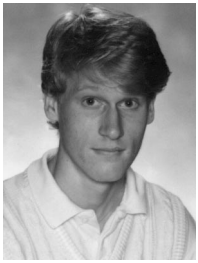
One may easily become convinced that the mean and median can be defined for any data set, the members of which are related by some distance function. The proposed algorithms can thus be applied more generally than to sequential data only.

References

[1] T. Kohonen, Content-Addressable Memories, Springer Series in Information Sciences, vol. 1, Springer, Heidelberg, 1980.
[2] T. Kohonen, Self-Organization and Associative Memory, Springer Series in Information Sciences, vol. 8, Springer, Heidelberg, 1984.
[3] T. Kohonen, Median strings, Patt. Rec. Lett. 3 (1985) 309.
[4] T. Kohonen, Self-Organizing Maps, Springer Series in Information Sciences, vol. 30, Springer, Heidelberg, 1st ed., 1995; 2nd ed., 1997.
[5] T. Kohonen, Self-organizing maps of symbol strings, Report A42, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland, 1996.
[6] T. Kohonen, J. Hynninen, J. Kangas, 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] T. Kohonen, E. Reuhkala, A very fast associative method for the recognition and correction of misspelt words, based on redundant hash addressing, in: Proc. 4th Int. J. Conf. on Pattern Recognition, Kyoto, Japan, 7–10 November 1978, pp. 807–809.
[8] T. Kohonen, H. Riittinen, E. Reuhkala, S. Haltsonen, On-line recognition of spoken words from a large vocabulary, Information Sciences 33 (1984) 3.
[9] T. Okuda, E. Tanaka, T. Kasai, A method for the correction of garbled words based on the Levenshtein metric, IEEE Trans. Comput. 25 (2) (1976) 172.
[10] B. Oommen, E. de St. Croix, String taxonomy using learning automata, IEEE Trans. Syst. Man Cybernet. B 27 (2) (1997) 354.
[11] J. Sammon Jr., A nonlinear mapping for data structure analysis, IEEE Trans. Comput. 18 (5) (1969) 401.
[12] K. Torkkola, J. Kangas, P. Utela, S. Kaski, M. Kokkonen, M. Kurimo, T. Kohonen, Status report of the Finnish phonetic typewriter project, in: T. Kohonen et al. (Eds.), Artificial Neural Networks, vol. 1, Elsevier, Amsterdam, 1991, pp. 771–776.

Teuvo Kohonen is a Professor of the Academy of Finland and head of the Neural Networks Research Centre, Helsinki University of Technology, Finland. He received his D.Eng. degree in physics from Helsinki University of Technology, Espoo, Finland in 1962. His research interests are in associative memories, neural networks, and pattern recognition. He is the author of five textbooks, including *Self-Organization and Associative Memory* (Springer, 1984) and *Self-Organizing Maps* (Springer, 1995). Over 3000 articles have been written worldwide on his self-organizing map (SOM) algorithm and its variations.

Dr. Kohonen has received many internationally acknowledged awards and prizes. He is also an honorary doctor of the University of York, United Kingdom, University of Dortmund, Germany, and Åbo Academy, Finland. He is a member of the Academia Scientiarum et Artium Europaea, the Académie Européenne des Sciences, des Arts et des Lettres, the Finnish Academy of Sciences, and the Finnish Academy of Engineering Sciences. He served as President of the European Neural Network Society and was the First Vice Chairman of the International Association for Pattern Recognition.



Panu Somervuo received his M.Sc. degree in Electrical Engineering from Helsinki University of Technology in 1996. He works as a research scientist at the Neural Networks Research Centre, Helsinki University of Technology. His research interests include Self-Organizing Maps and speech recognition.