neighborhood $NE = 0$ for 5 iterations, the resulting segmentation is presented in Fig. 6.

5	1	3	42	95	**582**
0	0	4	38	150	213
2	7	42	**652**	291	37
6	8	21	51	14	**787**
2	10	26	97	36	18
4	2	22	14	**793**	21

Table 2. The matching scores of the AR map

Fig. 6. Segmentation result by reduced 4×1 map

4. Conclusions

A novel method has been presented, based on a self-organizing map of AR models, that is capable of dividing a 1D or 2D signal into temporal or spatial segments, each of which can be modeled as a different AR process. The method is totally unsupervised and automatic and requires no interaction by the user nor any a priori knowledge of the signal.

We have demonstrated the use of the method for segmentation of an image into different textures. Other possible applications, that are not investigated here, are recognition of speech and other similar nonstationary time series, like seismic signals and biomedical signals, and identification of dynamic systems that change between several different processes in unknown intervals.

5. References

1. Brodatz, P., "Textures: A Photographic Album for Artists and Designers", Reinhold, New York, 1968.
2. R.L.Kashyap, R.Chellappa, and A.Khotanzad, "Texture classification using features derived from random field models", Pattern Rec. Letters 1, 1982, pp. 43-50.
3. R.L.Kashyap and K.B.Eom, "Texture boundary detection based on the long correlation model", IEEE Trans. PAMI 11, no. 1, 1989, pp. 58-67.
4. T. Kohonen, "Self-Organization and Associative Memory", Springer-Verlag, Berlin, Heidlberg, 1984.
5. T. Kohonen, "The "neural" phonetic typewriter", Computer 21, no. 3, 1988, pp. 11-22.
6. N.M.Nasrabadi and Y. Feng, "Vector quantization of images based upon the Kohonen self-organizing feature maps", Proc. IEEE Int. Conf. on Neural Networks, San Diego, USA, July 24-27, 1988, pp. 101-108.
7. B. Widrow, P.E.Mantey, J.L.Griffiths, and B.B.Goode, "Adaptive antenna systems", Proc. IEEE 55, no. 12, 1967, pp. 2143-2159.

all the surrounding pixels, then the same unit will match the center pixel even with twice as large an estimation error than any other unit. This neighborhood pressure makes possible to use maps with a smaller number of units, because the response does not spread on very many units. In Fig. 4. the map of the previous example was iterated with $C = 8$ for 50 iterations, and the effect is noticeable. As the units adapt better to the processes present in the image, the neighborhood pressure can be increased by decreasing C. Reducing C to 1 during 50 iterations gave rather good results, as can be seen from Fig. 5.

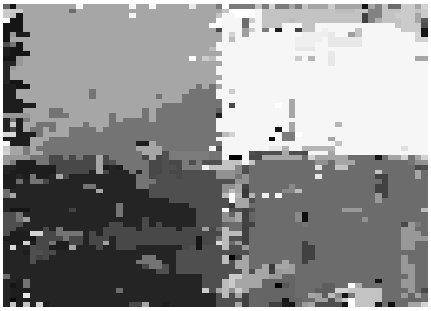

Fig. 4. Segmented test image with neighborhood pressure $C = 8$

Fig. 5. Final result after 50 iterations and $C = 1$

The segmentation result in Fig. 5. is not the same as a human would achieve. For example, the upper left texture is divided into two segments, which looks like an artefact at first glance, but actually there is a small change in the spacing between the lines.

If different areas in a texture have different AR models, there will be different units matching the areas. The similarity of the areas is reflected on the map as the close distance between the respective matching units, and this can be used to give the same label to all the units corresponding to similar AR models. Namely, if several nearly identical AR units match to parts of the same texture, and they are near each others on the map, then in a 2D array of matching scores of the units there will be a local maximum. The number of such local maxima is the number of clearly different textures in the image, and all the units belonging to the same local maximum detect the same texture. If variations inside the textures are much smaller than those between the different textures, then it is possible to automatically label all the units around each local maximum by the same label, giving rather good results.

However, if the map is small or there are similar textures, the local maxima will be rather near to each other, and it is difficult to automatically cluster the units between two maxima. An easier approach is then to pick up just the best unit from each maximum and make a new (not topological) map, that has exactly one unit for each texture. This reduction can be done as soon as there are stable local maxima in the matching frequencies, and a substantial number of all the pixels are concentrated at these maxima.

The matching scores of the 6×6 map after training (the case of Fig. 4.) are presented in Table 2. There are four clear local maxima, presented by boldface, suggesting that there are four textures in the image. After fine tuning the new map of those units with

eliminate these problems we scanned the image in random direction one row or column at a time. The history of a sample was naturally kept the same independent of the scanning direction, and it was the half-cross: half of the samples to the left of the predicted pixel and another half upwards. In fact, if the scanning occurs from right to left, the model is now anticausal.

The size of the map was found to have a rather small effect on the result as long as there are enough units, since the response will eventually localize to just a few units anyway. An example of the result with a 6×6 map with order of AR predictors 8 is shown in Fig. 2. Each pixel of Fig. 1. is replaced by a gray scale color representing the best matching unit for that pixel. The mapping of units to colors is shown if Fig. 3, which is an image of the map with the units replaced by their colors. The neighborhood was 2 at the beginning and decreased to 0 after 100 iterations over the test image.

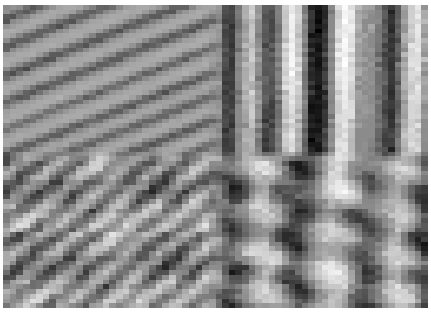

Fig. 1. The test image

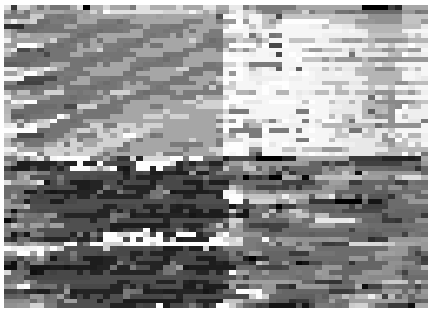

Fig. 2. Response of a 6×6 2^{nd} order map to the test image

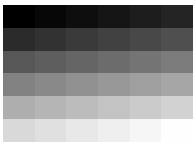

Fig. 3. The colors of the units

The result is rather noisy, due to the noisy original image, but clearly the different textures have responses at different units on the map, and the response of each texture is localized to a continuous area on the map. In the optimal case the responses of different textures would not overlap on the map, so that each unit would match only to pixels from one texture. These four textures, however, are not exact AR processes, and there are inevitable prediction errors that make the response to spread over an area of units. Note that, since training is totally unsupervised, the map in fact finds a set of "microtextures" of order 8 determined by the spatial geometry of the history vector. The microtextures correspond to parts of the four large textures.

Most of the actual errors are solitary pixels or short lines inside a homogenous area, thus they cannot represent any real texture segment, since a texture, by definition, is a pattern of several pixels. We used a heuristic approach to discourage the units getting tuned to solitary pixels by favoring those units, that also match the neighbors of the current pixel in the image. The averaged estimation error in eq. (5) was scaled by a factor that describes how homogenous the texture is:

$$v_i' = \frac{\beta e_i(n) + (1-\beta)v_i}{C + Nc}, \tag{6}$$

where the constant C determines the weight of the neighborhood pressure and Nc is the number of pixels in the 8-neighborhood of the current pixel, for which the unit i has been selected as the best matching unit. With $C = 8$, for example, if one unit matches best

shows the state of the map (frequency and distance from zero of the associated pole). Each point in the 128 point training signal was matched to some unit, and Table 1. also shows the distribution of these matches.

Unit	0	1	2	3	4	5	6	7
Frequency	1.789	2.212	3.142	1.157	1.121	1.100	0.891	0.492
Pole dist	0.999	0.912	1.049	0.907	1.005	1.000	0.938	0.987
Matches	34	1	2	0	31	29	0	31

Table 1. State of a 2^{nd} order 8×1 map and distribution of the test samples.

There are four units detecting each of the input frequencies, and some intermediate units that predict no samples at all. Within every test signal there are 1 or 2 samples that have been matched by a wrong unit, because on signal boundaries it is impossible to predict correct values with history belonging to another signal, and the discontinuities on the boundaries make false matches to the high frequency units 0, 1 and 2. The important point from Table 1 is that the map is able to separate the close frequencies 1.10 and 1.12 . At the beginning of the iteration both signals match to the same unit, which will have a pole between the two frequencies well inside the unit circle, but since the neighbors of the unit are updated also, there will be several units detecting almost the same process. When the neighborhood is decreased, the response will split to two units.

Tests with higher order 1D artificial AR processes resulted in the same typical behavior: the self-organizing map can find out all the AR processes in the input signal, but if the orders of the processes vary much, then the same unit may adapt to more than one lower order process. This can be controlled by the size of the neighborhood and the averaging parameter β in Eq. (5).

The tests with the artificial data indicate that self-organization is a useful concept with AR models, and the self-organizing AR map can separate rather similar processes without any prior knowlegde of the number of processes or the segment boundaries.

3.2. Self-Organization of AR models for 2D Textures

The main purpose of the study was to investigate whether the self-organizing AR maps can be used for automatic segmentation of an image into different textures. The test image was constructed from four texture samples, three Brodatz textures [1], numbers 16, 49 and 77, and an image of handwritten lines on a paper. The images were recorded by a poor quality video camera and a 32×32 sample was extracted from each of the four images. The samples had rather large natural noise and degradation. The composed 64×64 test image is shown in Fig. 1.

The original Kohonen self-organizing map [4] requires that the input samples are in random order. In this case, however, the samples come from a continuous process, thus the order of the samples is an essential part of the information. Scanning the image row by row is not very good since the textures have irregularities and noise which require rather small averaging factor for the prediction error ($\beta \sim 0.15 \cdots 0.30$ in Eq. (5)), and this will shift the response to texture boundaries in the direction of scanning. To

correlation with the sample to be predicted, but it is of course data dependent. The lower the order of the AR model is, the faster the *alms* method will converge. On the other hand, the dominant frequencies of real world textures vary within a large range, and in the test images used in this study were often rather low, which suggests a long spatial history. This is of course totally dependent on the relative spatial pixel resolution of the texture image. The best compromise was found to be a half-cross shaped history, with half of the samples to the left of the predicted pixel on the same image row and another half upwards on the same column.

Since the image is not actually a time series, there is no reason to use a causal AR model. Instead, the history could be symmetric around the predicted sample, and this would be essential if the texture segments in the image are small. Then the causal half-cross shaped history will inevitably give erroneous results at upper left corners of rectangular segments, since the whole history is from different textures than the pixel to be predicted. However, the convergence of the *alms* gets slower when the order of the AR model decreases, and in our test image the texture segments were rather large compared to the size of the history. In the following 2D examples we used a non-symmetric half-cross shaped history.

3. Results

3.1. Self-organization of 1D models with artificial data

The concept of self-organization of AR models was first explored with simple artificial data. A digital 128 point long signal was constructed, which consisted of 4 consequent 32 point long segments. Each segment was a sinusoidal signal with frequencies 0.5, 0.9, 1.6 and 2.2, respectively. In the first experiment the map consisted of four second order AR predictors in a linear lattice. With a neighborhood of $NE = 1$ the map was organized after less than 5 iterations over the input signals, which means that there is one unit detecting each of the input frequencies, and the frequencies appear in the lattice in either ascending or descending order. The poles of the AR models were all initially near the zero, but moved quickly to correct frequencies. Due to the neighborhood effect the AR models had first rather wide passband, but as the neighborhood shrinks, the poles move nearer to the unit circle and the final convergence is the same as if the *alms* had been applied to each signal separately.

Adding noise to the signals made the convergence slightly slower, when compared to separate *alms* for each signal, since the noise makes random errors to the prediction and wrong units get updated. With 20% white uncorrelated uniform noise added, the number of iterations required to the same convergence (poles at the same distance from the unit circle and correct frequencies) was increased by a factor of 2. Even with 100% of added noise, with signal to noise ratio 1.0, the map was able to converge although very slowly.

To investigate the capabilities of self-organization in this context we set up a more difficult test: the frequencies of the consequent segments in the input signal were 0.5, 1.10, 1.12 and 1.8 with additive 4% uniform noise. The lattice had 8×1 units of order 2 . The neighborhood was initially 5 and decreased to zero after 1000 iterations over each of the input signals (total number of points iterated was thus 128 000). Table 1.

Eq. (2) by using its own weight vector. Since the units are tuned to different AR models, some of the errors (3) will be smaller than the others. The apparent winner (the best matching unit) for each input vector $X(n)$ is the unit with the smallest estimation error. However, real world signals are noisy and error-free estimation with an AR model is not possible. The estimation error is noisy as well. For optimum weights, the time average of the estimation error is minimum, but at any given moment there is a large probability of a wrong unit giving the smallest error due to noise. Therefore, to measure the matching goodness of a unit i, an exponential average over the recent estimation errors has to be computed:

$$v_i' = \beta e_i(n) + (1 - \beta)v_i, \tag{5}$$

where β is a smoothing factor. Now the best matching unit is the one with the smallest v_i'. The value of β determines how long the same unit tends to keep active, and it depends of the size of segments desired.

Once the best-matching unit has been found, that unit and its neighbors are updated according to the *alms* algorithm (4). To define the neighborhood in the 2D lattice, the distance r of two units in the lattice must be defined. Neighborhoods determined by both the Euclidean and the Chebyshev metrics have been tried and they give similar results. When the neighboring units are updated, the scalar adaptation constant γ of Eq. (4) is multiplied by a factor depending on r to yield

$$g(r) = \gamma(1 - \frac{r}{NE + 1}), \tag{6}$$

where r is the distance from the best matching unit to the unit to be updated and parameter NE determines the size of the neighborhood. The size is slowly decreasing in time, e.g., if t denotes the iteration step, then

$$NE(t) = NE(0)(1 - \frac{t}{t_{max}}), \tag{7}$$

where t_{max} is a predetermined number of training steps. The coefficient γ in (6) is also slowly decreasing according to a similar law as NE. The summary of the algorithm is thus:

0) Initialize all coefficient vectors W_i of the units to small random values.
1) Input a sample vector $X(n)$ and the true value $x(n + 1)$. Compute the average error v_i' for every unit from Eqs. (5) and (3). Find the best matching unit with the smallest error.
2) Update the best matching unit and its neighborhood according to

$$W_i' = W_i + g(r)e(n)X(n)$$

and go to step 1).

If the signal is two dimensional, the only difference is the construction of the "history" vector $X(n)$. If the image is scanned rowwise, the actual causal history is a set of all the pixels to the left and up from the current pixel, and there are several ways to select the spatial history. The optimal history would the the set of pixels that have the largest

In Section 2, the learning algorithm is given, and in Section 3, results are given for two segmentation problems: first, the feasibility of the method was tested on artificial 1D autoregressive processes, then an image consisting of four natural textures was segmented.

2. Self-Organizing AR models

For simplicity, consider first the case in which the neural units learn to represent autoregressive (AR) models for a 1D input signal. Let $x(k), k = 0, 1, \ldots$ be a discrete time sequence (time series) obeying an AR model

$$x(n+1) = w_0 x(n) + w_1 x(n-1) + \ldots + w_{M-1} x(n-(M-1)) + e(n) \qquad (1)$$

The order of the AR model is M, the AR weights are the numbers w_0 to w_{M-1}, and $e(n)$ is the error term. Denote the vector of weights by W, and the time history of the current input sample $x(n+1)$ by $X(n)$. Thus $X(n)$ consists of samples $x(n)$, $x(n-1), \ldots, x(n-(M-1))$. Both W and X are column vectors of length M. If the $x(n)$ are samples from a noiseless AR process, the next sample can be predicted with no error from the M previous samples:

$$x(n+1) = X(n)^T W \qquad (2)$$

It is well-known that a second order AR model estimates correctly a sinusoidal sequence with one frequency. In general, an AR model of order M can correctly estimate a signal with $M/2$ frequencies, since in the z-plane an AR model has always symmetric poles with respect to the real axis.

The optimal weights for an AR model can be estimated from observations by writing down the estimation errors

$$e(n) = x(n+1) - X(n)^T W \qquad (3)$$

and computing the weights minimizing the errors in the least mean square sense (Yule-Walker equations). For our purposes, a more suitable algorithm is the adaptive least mean square method, *alms* , which is just one form of the well known Widrow - Hoff rule (see, e.g., [7]). After every input sample, the new weight vector W' is

$$W' = W + \gamma e(n) X(n), \qquad (4)$$

where γ is a scalar adaptation constant and $e(n)$ is the estimation error from Eq. (3).

The self-organizing map of AR models is a 2D lattice of units, indexed in the following by i, with weight vectors W_i each signifying an AR process. According to Kohonen [4], the general procedure for self-organization in the lattice is:

1) For each input sample, find the best matching unit
2) Modify that unit, and the units in its topological neighborhood, to further improve the match to the present input.

At a given step of the self-organizing algorithm, the inputs to the network, shared by all the units, are the samples in vector $X(n)$. Each unit tries to predict $x(n+1)$ from

Self-Organizing Maps for Spatial and Temporal AR Models

Lampinen J. and Oja E.

Lappeenranta University of Technology, Department of Information Technology
PO Box 20, SF-53850 Lappeenranta, Finland

Abstract

A method is given for unsupervised segmentation and classification of 1D and 2D signals. The method is based on a self-organizing map of "neural" units, like Kohonen's feature map, where every unit represents an AR model with its reference vector. The map will self-organize during an unsupervised learning phase. Several training segments of the signals are presented to the map, and each unit will learn to model different parts of the signals. The results indicate that the self-organizing AR map can learn to distinguish textures from images with unsupervised learning, which makes it suitable for segmentation of an image into different texture classes.

1. Introduction

In problems like speech recognition and scene analysis, it is essential to be able to segment the 1D waveform or 2D image into meaningful regions. Often this is achieved through supervised classification methods. However, in some cases there is little or no a priori knowledge about the number of different classes or the characteristics of the classes, and unsupervised segmentation methods are needed. The present paper gives such a method which is based on modelling the signals locally as autoregressive stochastic processes and clustering the parameter vectors by a neural network technique.

The main purpose of this study was to investigate the unsupervised segmentation of an image into texture classes. Autoregressive models have been used for texture classification and synthesis [2]. Recently, the use of random field model for the unsupervised segmentation problem was demonstrated [3]. In the present study no a priori knowledge about the number of classes or the class models is used. Instead, an artificial learning neural network is used with a predetermined number of "neural" units, each of which automatically learns to detect one typical texture model appearing in the input field. The network is the self-organizing topological map introduced by Kohonen [4].

This is a network of simple neural units arranged to a regular 2-dimensional lattice. In a learning algorithm, each unit becomes sensitive to a small part of the input space. Neighboring units tend to detect similar inputs which introduces a specific topological order of the inputs on the lattice. The topological map has been earlier applied to some problems in Pattern Recognition and Image Processing, e.g. to speech recognition [5] and to vector quantization of images [6].

Work on this paper was supported by Tekes Grant 4196/1988 under the Finsoft project.