
PERGAMON

Neural Networks 15 (2002) 979–991

Neural Networks

www.elsevier.com/locate/neunet

2002 Special Issue

Recursive self-organizing maps

Thomas Voegtlin*

Institut des Sciences Cognitives, CNRS UMR 5015, 67 boulevard Pinel, 69675 Bron Cedex, France

Abstract

This paper explores the combination of self-organizing map (SOM) and feedback, in order to represent sequences of inputs. In general, neural networks with time-delayed feedback represent time implicitly, by combining current inputs and past activities. It has been difficult to apply this approach to SOM, because feedback generates instability during learning. We demonstrate a solution to this problem, based on a nonlinearity. The result is a generalization of SOM that learns to represent sequences recursively. We demonstrate that the resulting representations are adapted to the temporal statistics of the input series. © 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Recursive self-organizing maps; Kohonen map; Recursiveness; Recurrent networks; Time

1. Introduction

There have been a number of approaches to representing time in neural networks, often related to language and speech recognition. The crucial distinction which separates these approaches is the nature of the representation. In an *explicit* representation, time is considered as a dimension of space. This is the case when delays are used to collect different measures of an input over a time window (Lang, Waibel, & Hinton, 1990). In an *implicit* representation, time is represented indirectly, by the effect it has on processing (Elman, 1990). This is the case with recurrent networks, or with so-called 'leaky integrators' units. In general, explicit representations of time are limited and oversensitive to temporal deformations of the signal, while implicit representations are powerful and more robust to deformations. A limitation of leaky integrators neurons is that long-term information tends to decay exponentially. Recurrent representations, however, do not necessarily have this limitation[1] and they are conceptually more appealing.

Various approaches have been proposed for representing time with the self-organizing map (SOM). This includes explicit representations (Kangas, 1994; Vesanto, 1997), methods based on lateral connectivity, on recurrent connections (Euliano & Principe, 1996; Hoekstra & Drossaers, 1993; Kopecz, 1995), on leaky integrators (Chappell & Taylor, 1993; Privitera & Morasso, 1994), or on combinations of those principles (James & Miikkulainen, 1995; Koskela, Varsta, Heikkonen, & Kaski, 1998; Mozayyani, Alanou, Dreyfus, & Vaucher, 1995). A common theme in these proposals is that they attempt to generalize self-organization to time. However, it is not clear in what sense the properties of SOM should be generalized to time. In the evaluation of time-enhanced SOMs, classical notions like quantization error or distortion have not been used so far, because they do not directly apply.

In this paper, we present recursive SOM, a modified SOM where feedback is used to represent time. The representation is implicit and self-referent, in the sense that the map learns to classify its own activities. In this aspect, our approach differs from most models proposed so far. However, it is based on the very natural idea of adding recurrent connections to SOM, while preserving its original self-organization principle. In addition, the proposed approach is compatible with related algorithms, and therefore it has some generality; as an illustration, we present a similar extension of the Neural Gas algorithm (Martinetz & Schulten, 1991). In order to assess performance, we propose a generalization of quantization error to time series. We demonstrate that classical vector quantization properties of SOM are successfully generalized to time.

2. Self-reference and self-organization

A well-known example of an implicit representation of time in a recurrent network is the simple recurrent network (SRN) by Elman (1990). The SRN is a modified perceptron with one hidden layer that uses a time-delayed copy of its hidden layer activities as additional input. The back-propagation learning algorithm is applied to both the input

* Tel.: +33-4379-11265; fax: +33-4379-11210.
E-mail address: voegtlin@isc.cnrs.fr (T. Voegtlin).

[1] Unless gradient methods are used, see Bengio, Simard, and Frasconi (1994) for details.

PII: S0893-6080(02)00072-2

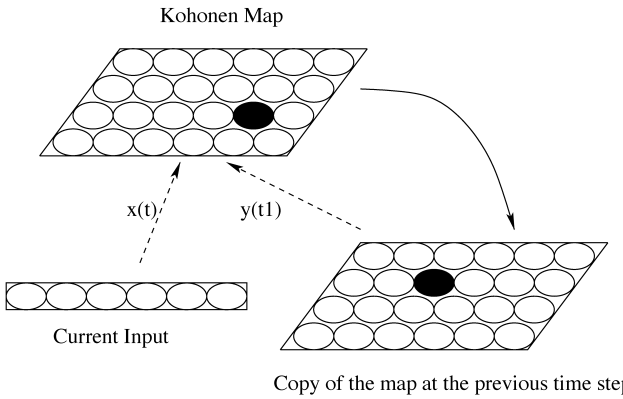

Fig. 1. Recursive SOM architecture. The original SOM algorithm is used recursively, on both the input vector $\mathbf{x}(t)$ and the representation derived at the previous time step, $\mathbf{y}(t-1)$. Dotted lines represent trainable connections. The continuous arrow represents fixed one-to-one connections. The network learns by associating current input to previous activity states. Hence, each neuron responds to a sequence of inputs.

vector and this time-delayed copy. Since the SRN learns from its own past activities, the representation in its hidden layer is self-referent.

In a self-referent representation, feed-forward and feedback connections are homogeneous; there is no difference between them in the equations that describe neural activities, nor in the learning rules. It is in this sense that we propose to use self-reference with the SOM, by adding recurrent connections to the original architecture. Although other temporal extensions of SOM have been based on recurrent connections (Euliano & Principe, 1996; Hoekstra & Drossaers, 1993; Kopecz, 1995; Koskela et al., 1998), in these extensions feed-forward and feedback connections are not homogeneous. In contrast, our model uses a time-delayed copy of its activities, and has homogeneous connections (Fig. 1). Both the input and the time-delayed copy of activities are considered as one single input vector to the classical SOM algorithm. Hence, each unit of the map will learn to represent a couple (input, context). Representations of long sequences are learned iteratively, based on previously learned representations of shorter sequences.

In the original SOM (Kohonen, 1989), each unit i of a map ($1 \leq i \leq N$) compares its weight vector $\mathbf{w}_i$ to an input vector $\mathbf{x}(t)$, where t denotes the time index. In general the Euclidean norm is used in the comparison. For a given input vector, the best matching unit is the unit that minimizes the quantization error $E_i = \|\mathbf{x}(t) - \mathbf{w}_i\|$. The learning rule updates the weights of neurons that belong to a neighborhood of the best-matching unit, in the direction of the input vector:

$$\Delta\mathbf{w}_i = \gamma h_{ik}(\mathbf{x}(t) - \mathbf{w}_i) \tag{1}$$

where k is the index of the best-matching unit, γ a learning rate and h_{ik} is a so-called neighborhood function, that decreases with the distance between units i and k on the map.

We use the same notations in the description of our algorithm. In addition, let $\mathbf{y}(t)$ denote the vector of activities in the map at time t. Each unit i of the map ($1 \leq i \leq N$) has two weight vectors, $\mathbf{w}_i^x$ and $\mathbf{w}_i^y$, that are compared to the input vector $\mathbf{x}(t)$, and to the activities at the previous time step, $\mathbf{y}(t-1)$, respectively. In order to select a best-matching unit, it is necessary to combine the quantization errors corresponding to $\mathbf{x}(t)$ and $\mathbf{y}(t-1)$. Since we want feed-forward and feedback connections to be homogeneous, we sum the squared quantization errors:

$$E_i = \alpha\|\mathbf{x}(t) - \mathbf{w}_i^{\mathbf{x}}\|^2 + \beta\|\mathbf{y}(t-1) - \mathbf{w}_i^{\mathbf{y}}\|^2 \tag{2}$$

with $\alpha > 0$ and $\beta > 0$. The role of parameters α and β will be explained later. The best-matching unit is defined as the unit that minimizes E_i. If $k = \arg\min\{E_i\}_{1\leq i\leq N}$, then the learning rules used for updating feed-forward and recurrent weights are:

$$\Delta\mathbf{w}_i^{\mathbf{x}} = \gamma h_{ik}(\mathbf{x}(t) - \mathbf{w}_i^{\mathbf{x}}) \tag{3}$$

$$\Delta\mathbf{w}_i^{\mathbf{y}} = \gamma h_{ik}(\mathbf{y}(t-1) - \mathbf{w}_i^{\mathbf{y}}) \tag{4}$$

where γ is a learning rate, and h_{ik} is a neighborhood function. Note that this is simply the original SOM learning rule applied to both vectors $\mathbf{x}(t)$ and $\mathbf{y}(t-1)$.

So far the definition of our model has been straightforward, resulting from the idea of using feedback in a SOM. However, in order to complete this definition, we need to relate the neural activities $\mathbf{y}(t)$, to the input and synaptic weights. Let y_i denote the ith component of vector $\mathbf{y}$. The activity y_i of unit i will depend on the squared quantization error E_i. However, using $y_i = E_i$ directly would result in unstable representations. It is necessary to use a transfer function, $y_i = F(E_i)$. The reason why, as well as the choice of an appropriate transfer function will be explained in the following sections.

Note that parameters α and β can be removed from the definition of E_i; it is equivalent to use $E_i' = \|\mathbf{x}(t) - \mathbf{w}_i^{\mathbf{x}}\|^2 + \|\mathbf{y}(t-1) - \mathbf{w}_i^{\mathbf{y}}\|^2$ and $y_i = \sqrt{\beta/\alpha}F(\alpha E_i')$. Therefore feedback and feed-forward connections are homogeneous in our algorithm. We have included α and β in E_i because it facilitates the analysis of a transfer function. However, α and β may be considered as part of the transfer function.

3. Considerations on stability

The idea of using feedback with a SOM is not new. Previous SOMs with feedback have been proposed by Briscoe and Caelli (1997) and Scholtes (1991). However, an important problem with feedback in SOM has been instability during learning (Scholtes, 1991). Typically, when the order of the representation changes on the map, the feedback vector changes violently, which prevents stable representations from being learned. The results presented by Briscoe and Caelli (1997) suggest that stability is not achieved in their model. In the present section, we present an approximate stability analysis of recursive SOM.

Although approximate, this analysis will help understand the choice of a transfer function.

Consider a small perturbation $\delta\mathbf{y}(t)$ of the feedback vector $\mathbf{y}(t)$ at time t. We assume that all other variables (the input sequence, the values of synaptic weights) are unchanged. Perturbation $\delta\mathbf{y}(t)$ will have two effects. First, the next values of the feedback vector, $\mathbf{y}(t+n)$, for $n \geq 0$, will be directly modified by recurrent propagation of modified activities. Let $\delta\mathbf{y}(t+n)$ denote this perturbation. Secondly, a modification of the distribution of $\mathbf{y}$ may induce long-term adaptation of the synaptic weights through learning. For the moment, however, we only consider short-term dynamics. If the learning rate is small enough, adaptation of the weights can be neglected at first.

Short-term stability will depend on the evolution of $\delta\mathbf{y}(t+n)$. If $\|\delta\mathbf{y}(t+n)\| \to 0$ when $n \to \infty$, then the perturbation will vanish. Moreover, if $\sum_{n\geq 0}\|\delta\mathbf{y}(t+n)\| < +\infty$, then the distribution of $\mathbf{y}$ will remain unchanged, and no long-term adaptation of the synaptic weights will occur. In addition, it is necessary to ensure that the effect of a perturbation is small if this perturbation is small. This can be expressed using the Landau notation O, in the following condition:

$$\sum_{n\geq 0}\|\delta\mathbf{y}(t+n)\| = \mathrm{O}(\|\delta\mathbf{y}(t)\|) \tag{5}$$

Now let us consider the effects of weights adaptation on stability. To be quite rigorous, stability of the weights should be considered only when the learning rate decreases to zero. However, we consider stability in a broad sense here, that is, we assume that the learning rate is constant and small, and we consider convergence of the expected values of the synaptic weights; the variance of the weights during learning can be made arbitrarily small by the choice of a learning rate. To keep things simple, we also assume that weights converge in this broad sense when the classical SOM algorithm is used and the distribution of the input vector is constant.[2]

Let $\mathbf{W}^{\mathbf{y}} = (\mathbf{w}_i^{\mathbf{y}})_{1\leq i\leq N}$ denote the weights matrix of the recurrent connections. Consider a small modification $\delta\mathbf{W}^{\mathbf{y}}$ of $\mathbf{W}^{\mathbf{y}}$ during learning. As mentioned above, we consider that the elements of $\mathbf{W}^{\mathbf{y}}$ are the expected values of the weights during learning, and that $\delta\mathbf{W}^{\mathbf{y}}$ is a modification of these expected values. The modification $\delta\mathbf{W}^{\mathbf{y}}$ will induce a direct modification of $\mathbf{y}(t)$, because $\mathbf{w}_i^{\mathbf{y}}$ will be different in Eq. (2). In addition, $\mathbf{y}(t)$ may also be indirectly modified by propagation of previously modified activities, as seen above. Hence, $\delta\mathbf{y}(t)$ will be the result of a direct component and an indirect component. The direct component can be derived from Eq. (2), and will be the same order of magnitude as $\delta\mathbf{W}^{\mathbf{y}}$, assuming that F is continuous. The indirect component is a sum of terms, resulting from the perturbations at $t-1$, $t-2$, etc. However, if condition (5) is satisfied, then this sum will be the same order of magnitude as the direct component. Therefore, the overall $\delta\mathbf{y}(t)$ will be the same order of magnitude as $\delta\mathbf{W}^{\mathbf{y}}$:

$$\|\delta\mathbf{y}(t)\| = \mathrm{O}(\|\delta\mathbf{W}^{\mathbf{y}}\|) \tag{6}$$

Learning is a so-called 'moving target' problem, where the target is a set of synaptic weights that is stable during learning. The target is moving because the distribution of $\mathbf{y}$ changes when the weights of the recurrent connections are updated. However, if Eq. (6) is satisfied, then we may compare the speed of the weights to the speed of the moving target. Stability during learning will depend on this comparison; if the target moves faster than the weights, then the representation will be unstable. Alternately, if weights move faster than the target, then a stable representation of long sequences will progressively be learned. Hence, the question is whether it is possible to find a transfer function so that the expected weights move faster than the target.

Let $\mathbf{T} = (\mathbf{t}_i)_{1\leq i\leq N}$ denote the 'moving target' of recurrent connections, where each vector $\mathbf{t}_i$ is the 'target' of vector $\mathbf{w}_i^{\mathbf{y}}$. Each target vector $\mathbf{t}_i$ is defined as the value toward which $\mathbf{w}_i^{\mathbf{y}}$ would converge for a stationary joint distribution of $(\mathbf{x}(t), \mathbf{y}(t-1))$. Hence, the target at a given time results from the values of the synaptic weights and from the statistics of the input.[3] However, each change of the weights will modify the distribution of $(\mathbf{x}(t), \mathbf{y}(t-1))$ and therefore the target will move. In order to ensure that the expected weights move faster than the target, we need the modification of the target to be smaller than the modification of the weights:

$$\|\delta\mathbf{T}\| < \|\delta\mathbf{W}^{\mathbf{y}}\| \tag{7}$$

Due to local interactions in the map, each target vector $\mathbf{t}_i$ is a barycenter of the expected value of $\mathbf{y}(t-1)$ when unit i is selected at time t, and of the expected values of $\mathbf{y}(t-1)$ when neighbors of unit i are selected at time t. Although neighborhood relationships make the analysis of target perturbation difficult, it is reasonable to assume the following upper bound:

$$\|\delta\mathbf{T}\| \leq \sqrt{N}E[\|\delta\mathbf{y}\|] \tag{8}$$

where $\delta\mathbf{T}$ denotes the modification of the target matrix due to the perturbation of the distribution of $\mathbf{y}$, and $E[\cdot]$ denotes the statistical expectancy. Inequality (8) yields the following sufficient condition for Eq. (7):

$$\sqrt{N}E[\|\delta\mathbf{y}\|] < \|\delta\mathbf{W}^{\mathbf{y}}\| \tag{9}$$

We have derived two stability conditions. Condition (5) ensures that neural dynamics will be stable, and condition (9) ensures stability of the weights during learning. If both are satisfied, then the expected values of the weights should

[2] In fact, this has only been demonstrated for one-dimensional maps, see Kohonen (1997) for details. Here, however, we assume that this is true in the general case.

[3] For stationarity of the joint distribution of $(\mathbf{x}(t), \mathbf{y}(t-1))$, we assume that the input series $\mathbf{x}(t)$ is ergodic.

converge during learning. Now we may examine what constraints these conditions impose on the transfer function. The activity, $y_i(t)$, of unit i is:

$$y_i(t) = F(E_i) = F(\alpha\|\mathbf{x}(t) - \mathbf{w}_i^{\mathbf{x}}\|^2 + \beta\|\mathbf{y}(t-1) - \mathbf{w}_i^{\mathbf{y}}\|^2) \quad (10)$$

where F denotes the transfer function. First, we examine the evolution over time of a small perturbation $\delta\mathbf{y}(t+n)$ of $\mathbf{y}(t+n)$. For each $i \in \{1,\ldots,N\}$, we have:

$$(y_i + \delta y_i)(t+n) = F(\alpha\|\mathbf{x}(t+n) - \mathbf{w}_i^{\mathbf{x}}\|^2 + \beta\|(\mathbf{y} + \delta\mathbf{y})(t+n-1) - \mathbf{w}_i^{\mathbf{y}}\|^2) \quad (11)$$

A first-order Taylor approximation in $\delta\mathbf{y}$ yields:

$$\delta y_i(t+n) = 2\beta F'(E_i)(\delta\mathbf{y}(t+n-1))^{\mathrm{T}}\big(\mathbf{y}(t+n-1) - \mathbf{w}_i^{\mathbf{y}}\big) \quad (12)$$

where F' denotes the derivative of F, and superscript T denotes the transpose. Using the Cauchy–Schwarz inequality yields:

$$\frac{\|\delta\mathbf{y}(t+n)\|^2}{\|\delta\mathbf{y}(t+n-1)\|^2} \le \sum_{1\le i\le N} \big(2\beta F'(E_i)\|\mathbf{y}(t+n-1) - \mathbf{w}_i^{\mathbf{y}}\|\big)^2 \quad (13)$$

Similarly, a small change $\delta\mathbf{w}_i^{\mathbf{y}}$ of $\mathbf{w}_i^{\mathbf{y}}$ will result in a modification $\delta y_i(t)$ of the activity of unit i. A similar calculation yields:

$$\|\delta\mathbf{y}(t)\|^2 \le \sum_{1\le i\le N} \big(2\beta F'(E_i)\|\mathbf{y}(t-1) - \mathbf{w}_i^{\mathbf{y}}\|\big)^2 \|\delta\mathbf{w}_i^{\mathbf{y}}\|^2 \quad (14)$$

We may assume that the modification of the weights has the same amplitude for each column. That is, for each i, $\|\delta\mathbf{w}_i^{\mathbf{y}}\| = \|\delta\mathbf{W}^{\mathbf{y}}\|/\sqrt{N}$. This yields:

$$N\frac{\|\delta\mathbf{y}(t)\|^2}{\|\delta\mathbf{W}^{\mathbf{y}}\|^2} \le \sum_{1\le i\le N} \big(2\beta F'(E_i)\|\mathbf{y}(t-1) - \mathbf{w}_i^{\mathbf{y}}\|\big)^2 \quad (15)$$

Both inequalities (13) and (15) have the same right term. Hence, to achieve stability, it is sufficient to find a transfer function so that the expected value of this right term is strictly lower than one. In this case, condition (9) will be directly satisfied, and condition (5) will be satisfied as well, because any small perturbation $\delta\mathbf{y}$ of $\mathbf{y}$ will decay exponentially.

Note that a transfer function must take different values for the best-matching unit and for other units, or a trivial solution to the above inequalities would be $F = 0$. The fact that the derivative $F'(E_i)$ has to be bounded suggests that the transfer function needs to be continuous. For example, consider that the transfer function is a binary winner-take-all, so that the activity $y_k(t)$ of the winner unit k at time t is equal to one, and all the other units have activities $y_i(t)$ equal to zero ($i \neq k$). For a given input sequence, a small change of the weights might result in the selection of a different unit, and in this case the feedback vector will change violently, resulting in unstable representations.

4. Choice of a transfer function

We make the empirical choice of the following transfer function, for all $i \in \{1,\ldots,N\}$:

$$y_i = \exp(-\alpha\|\mathbf{x}(t+n) - \mathbf{w}_i^{\mathbf{x}}\|^2 - \beta\|\mathbf{y}(t+n-1) - \mathbf{w}_i^{\mathbf{y}}\|^2) \quad (16)$$

This function is continuous and takes on values between zero and one. Well-matching units will have activities close to one, while mismatching units will have activities close to zero. Intuitively, the activity of well-matching units should be stable when inputs, weights, or previous activities are perturbated, because it is a Gaussian function close to the origin. For mismatching units, activity is close to zero, and therefore it should be stable too.

We will demonstrate that stability conditions (5) and (9) can be satisfied with this transfer function. The right term of inequalities (13) and (15) is equal to:

$$\sum_{1\le i\le N} (2\beta\|\mathbf{y}(t-1) - \mathbf{w}_i^{\mathbf{y}}\|\exp(-\alpha\|\mathbf{x}(t) - \mathbf{w}_i^{\mathbf{x}}\|^2 - \beta\|\mathbf{y}(t-1) - \mathbf{w}_i^{\mathbf{y}}\|^2))^2 \quad (17)$$

The function: $z \to 2\beta z\exp(-\beta z^2)$ is bounded, and its maximum value is equal to $\sqrt{2\beta/e}$. In addition, $\exp(-\alpha\|\mathbf{x}(t) - \mathbf{w}_i^{\mathbf{x}}\|^2) < 1$. Although these might be very rough bounds, they are sufficient to demonstrate that for $\beta < e/(2N)$ stability is ensured. Therefore, stability conditions can be satisfied with the transfer function defined in Eq. (16). This choice completes the definition of our algorithm. Classical neighborhood functions may be used. In this paper, the neighborhood function is a Gaussian of the distance $d(i,k)$ between units i and k on the map, of width σ : $h_{ik} = \exp(-d(i,k)^2/\sigma^2)$.

In order to test our algorithm, we trained a two-dimensional recursive SOM of size 20×20 units on a corpus of written English, the novel "Brave New World" by Aldous Huxley. Letters were encoded on 5 bits, and presented to the network one at a time. Punctuation symbols were removed from the text and neural activities were reset to zero between words. We used a constant learning rate $\gamma = 0.1$, and the neighborhood function had a constant size $\sigma = 0.5$. Parameter values were $\alpha = 3$ and $\beta = 0.7$. During training, neurons became sensitive to sequences of letters. For each neuron, the receptive field is defined as the common end of all the sequences that trigger its selection. For instance, if a unit responds to 'nted', 'ited' and to 'red', then its receptive field is 'ed'. Receptive fields after 1000 presentations of the corpus are presented in Fig. 2. The presence of long receptive fields (e.g. 'their') demonstrates that learned weights are stable; stability is necessary for developing representations of long sequences. Topological

to	so	lo	o	ho	ie	me	me	me	ee	ne	ne	ne	it	lit	s	rs	aw	dow	tw
to	ro	co	wo	te	ite	e	e	e	lle	ine	ine	were	there		was	ss	sw	w	x
to	do	no	o	te	te	be	be	we	we	ore	re	re	re	is	was	ls		ew	ex
they	they		fo		se	se	are	see	we	here	re	are		his	es	us	rs	ds	ow
y	y	su		ev	pre	ru	pu	one	pe	pe		t	os	ins	es	ms	as	its	as
ey	my	pu	she	v	du	cou	nu	fu	ke	put	rat	that	ous	is	s	ts	ins	cons	as
ly	ly	ple	fe	ge	de	hu	ou	u	u	red	fort	ght	ns	res	s	fa	a	a	q
sh		le	e	the	de	mu	tu	bu	ed	red	ed	art	ot	is	ca	na	ba	ha	wa
sh	h	he	he	ce	ye	beg	hu	bu	ild	cond	ed	tt	att	et	t	ma	na	tha	ra
ch	wh	ple	one	whe	the	ng	ug	ag	g	nd	und	ut	rt	at	st	sc	c	ea	sa
th	h	with	e	ge	the	ng	g	ing	g	nk		but	not	at	nt	t	nc	a	ta
th	h	with	e	the	she	od	and	id	ad	k	k	int	ght	wit	et	an	an	n	wn
th	h	er	be	ve	ad	ted	id	ard	hed	el	out	t	it	it	wan	men	an	n	n
wer	er	er	der	rd	ther	had	ld	od	d	d	wl	t	int	on	when	han	then	ten	wen
ber	for	ter	rr	fr	r			tl		cl	cl	bel	all	on	in	fin	len	nn	un
or	her	tr	ar	str	dr	si	i	l	al	bl	sl	ll	all	lon	in	en	j	men	un
wer	wer	ur	r	br	z		ni	l	l	el	il	il	ul	om	ven	en	then	kn	con
pr	thr	their	ab	b	b	fi	ati	wi	hi	pi	ci	al	al	him	them	alp	up	sn	ep
ther	for	if		b	i	ri	ti	whi	di	bi	mi	al	sm	em	em		op	sp	ap
dr	af	f	of		hi	i	i	thi	li	li	ili	m	m	com	him	dl	ep	rep	p

Fig. 2. Receptive fields of a two-dimensional recursive SOM trained on English text. A receptive field is defined as the intersection of all the sequences that trigger selection of the corresponding unit. Receptive fields are displayed in natural reading order. Topographic organization is observed, principally based on most recent letters.

ordering can be observed in the placement of the receptive fields. In addition, the map has captured some statistical regularities of written English, as frequent groups of letters or short words are present in the receptive fields.

The above experiment demonstrates that it is possible to learn stable representations in recursive SOM. Stability is achieved by the choice of the transfer function defined in Eq. (16). However, the value of β we used was well beyond condition $\beta < e/(2N)$. This suggests that we can find better bounds than the above. First, we used the upper bound $2\beta z \exp(-\beta z^2) \leq \sqrt{2\beta/e}$ for all units. However, this function will reach its maximum for a number of units much less than N; it will be close to zero for mismatching units and for well-matching units. Only for 'not-so-well' matching units will it take on higher values. Secondly, using $\exp(-\alpha\|\mathbf{x}(t+n) - \mathbf{w}_i^{\mathbf{x}}\|^2) < 1$, does not take into account the influence of α. A much better approximation is to consider that the expected value of $\|\mathbf{x} - \mathbf{w}_i^{\mathbf{x}}\|^2$ is a constant, c, independent from the recurrent weights $\mathbf{W}^{\mathbf{y}}$ and from the unit index i; we may do this approximation because adaptation of the feed-forward weights uses the classical SOM algorithm, and is relatively independent from the recurrent weights. By making these assumptions, we obtain a stability condition of the type:

$$\beta < K \exp(2\alpha c) \tag{18}$$

where K is a constant. Here the critical value of β exponentially increases with α.

In order to test stability, we ran a series of experiments in which we observed the stability of a recursive SOM for different values of α and β. The map was of size 10×10 units, ($N = 100$), and the input vector was one-dimensional. The input series was the binary sequence generated by a simple two-state automaton. The two states were labeled '0' and '1'. The probability of transition from '0' to '1' was 0.3, and the transition from '1' to '0' had probability 0.4. Values of α and β were tested over the interval $[0.05, 2.00]$, using a step equal to 0.05. This yields 400 different conditions. For each condition, eight experiments were performed, starting with initial random weights. Maps were trained for 120 000 iterations, with constant learning rate and neighborhood size, $\sigma = 0.5$ and $\gamma = 0.3$, respectively. In order to assess stability, we measured the average length of the receptive field of the selected unit. Averaged results are presented in Fig. 3.

Fig. 3 shows that the plane (α, β) is partitioned into two domains. In one domain, the average length of the selected receptive field is below 2, which indicates instability during learning. In the second domain the average length of the selected receptive fields is between 5 and 7, suggesting stability. For $0 < \alpha < 1$, the transition area between stability and instability domains is relatively narrow, demonstrating that convergence occurs for values of β below a critical threshold. For values in this transition area, by examining the receptive fields during learning we observe stability for limited periods of time, followed by great reorganizations of the receptive fields. The value of the critical threshold increases with α; for $0 < \alpha < 1$, its progression compares well to the theoretical exponential progression of condition (18).

For $\alpha > 1$, the transition area, where the average length of the selected receptive field is between 2 and 5, becomes broader. This broadening of the transition area is due to poor performance when α and β are too high. However, by examining individual trials, we do not observe instability in this area. In fact, this poor performance is caused by the shape of the transfer function; increasing α and β increases the slope of the transfer function from $E_i' = \|\mathbf{x}(t) - \mathbf{w}_i^{\mathbf{x}}\|^2 +$

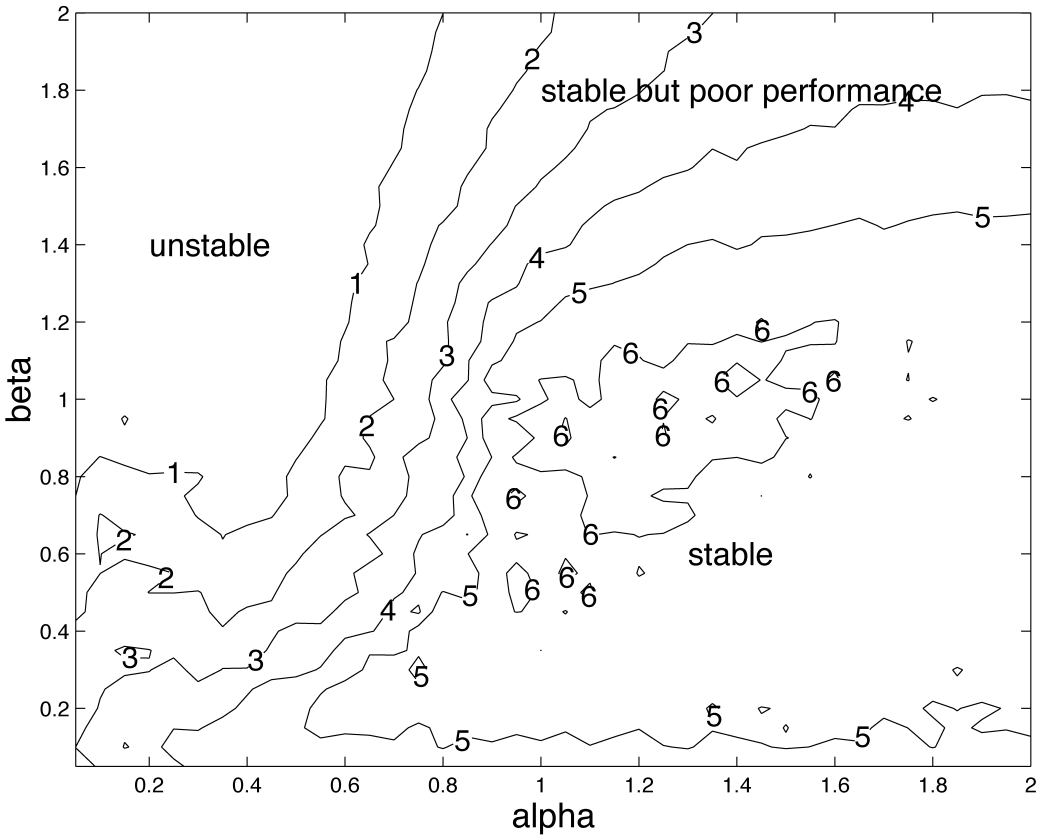

Fig. 3. Domain of stability of a recursive SOM trained on a binary sequence, for different values of parameters α and β. The mean length of the winner receptive field is displayed. For each condition, data are averaged over eight experiments. Low values indicate instability of the weights, and high values indicate stability. In the upper right region, stability is observed, although performance is relatively poor.

$\|\mathbf{y}(t-1) - \mathbf{w}_i^{\mathbf{y}}\|^2$ to $y_i = \sqrt{\beta/\alpha}F(\alpha E_i')$. If this slope becomes too high, then the mutual information between E_i' and y_i is reduced, which makes it more difficult to classify different pattern of activity. Hence, learning becomes increasingly difficult if α and β are too high.

5. Quantization of discrete sequences

Fig. 2 suggests that statistical regularities of written English are learned by the recursive SOM. In order to further investigate this point, we need a well defined measure of how well temporal regularities of the input series are captured. We do this by generalizing vector quantization and quantization error to time series.

Vector quantization consists in forming an inexpensive representation of a stochastic input vector. Formally, a *quantizer* is a function from the input space to a finite alphabet, or *code-book*, $L_1, \ldots, L_N$. A prototype vector, $\mathbf{w}_i$, is associated to each letter L_i of the alphabet; this prototype is used as a reconstruction of the input. A classical measure of the quality of a vector quantizer is the mean square quantization error, defined as $E[\|\mathbf{x}(t) - \mathbf{w}_{k(t)}\|^2]$, where $\mathbf{x}(t)$ is the input vector at time t, $k(t)$ the index of the prototype approximating $\mathbf{x}(t)$, and $E[\cdot]$ denotes the statistical expectancy.

Since recursive SOM learns a local representation of temporal context, it performs a quantization of temporal context. Hence, it should be possible to generalize quantization error to temporal context, in a way that reflects how well temporal sequences can be reconstructed from their representation. In fact, it will be necessary to consider different generalizations of quantization error, depending on whether the input vector takes on discrete or continuous values. In this section, we discuss the discrete case. A generalization to the continuous case will be presented in Section 6.

The input series consists of discrete, non independent, identically distributed (i.i.d.) vectors. For simplicity, we consider with no loss of generality that the input $\mathbf{x}(t)$ is one-dimensional and takes on binary values, and that the sequence of past events is infinite, so that the temporal context is an infinite sequence. Formally, the temporal context at time t is the sequence $c(t) = (\mathbf{x}(t), \mathbf{x}(t-1), \mathbf{x}(t-2), \ldots)$. Hence, $c(t) \in \{0,1\}^N$.

The temporal context is represented using an alphabet of N letters, $L_1, L_2, L_3, \ldots, L_N$, which plays the classical role of a code-book. At each time t, a unique letter, $L_{k(t)}$, represents $c(t)$. A reconstruction of $c(t)$ is performed, using the code-book as a set of references. Since the number of possible contexts is infinite, the reconstruction will be restricted to a finite number of past events. Hence, the quality of a

representation will be defined as the expected number of past events that can be correctly reconstructed.

Formally, we define a *context quantizer* as a function from the set of all possible contexts, $\{0,1\}^N$, to the codebook $\{L_1, L_2, L_3 \ldots, L_N\}$. The set of temporal contexts represented by L_i is called the *receptive field* of letter L_i, and it is denoted by R_i. The number of past events that can be unambiguously reconstructed from L_i will be denoted by n_i, and is called the *depth* of the receptive field R_i. n_i is the length of the intersection of all contexts belonging to R_i.

The set of all contexts, $\{0,1\}^N$, can be viewed as an infinite binary tree. The receptive field R_i of letter L_i is a set of binary sequences, that correspond to infinite paths starting from the root of this tree. A unique node H_i of the tree can be assigned to L_i, which is the deepest node of the tree that belongs to all the contexts in R_i. The depth of R_i is equal to the depth of node H_i in the tree. Since any context must belong to a receptive field, pruning the tree at nodes $H_1, \ldots, H_N$ results in a finite tree, whose leafs are $H_1, H_2, \ldots, H_N$. If all nodes $H_1, H_2, \ldots, H_N$ are distinct, then each receptive field R_i is equal to the set of paths passing through node H_i. In that case, the context quantizer is completely described by the finite binary tree $\{H_1 H_2 \cdots H_N\}$.

If p_i is the probability that context belongs to R_i, then the mean number of past events that can be reconstructed is:

$$\bar{n} = \sum_{1 \le i \le N} p_i n_i \tag{19}$$

$\bar{n}$ is called the *depth* of the quantizer. Depth is a quality measure of a quantizer, which reflects its memory of past events. In order to maximize depth, it is necessary to exploit the probabilistic structure of the sequence. Note that maximizing depth is the opposite of building a Huffman code of the input (Huffman, 1952). In Section 4, we have used depth as an indicator of stability.

Temporal extensions of SOM that build a local representations of sequences are context quantizers. Therefore it is possible to compare their performance by measuring depth. We present such a comparison here. Note that temporal extensions of SOM do not necessarily use local representations. An example of distributed representation, for which depth cannot be used, is SARDNET (James & Miikkulainen, 1995).

One of the first attempts to integrate temporal information in SOM has been the Temporal Kohonen Map (TKM) by Chappell and Taylor (1993). In TKM, the activity of a unit is defined as a function of earlier input vectors. Each unit i has a potential $V_i(t)$ at time t. The potential is updated according to

$$V_i(t) = dV_i(t-1) - (1/2)\|\mathbf{x}(t) - \mathbf{w}_i\|^2 \tag{20}$$

where $\mathbf{w}_i$ is the weight vector associated to unit i, and $0 < d < 1$ is a decay. The best-matching unit maximizes V_i, and the same learning rule as in the original SOM is used. It has been shown that TKM has strong limitations (Koskela et al., 1998), because weights converge toward linear combinations of the input patterns, resulting in limited representations. However, Varsta, Heikkonen, and Millan (1997) context proposed a powerful modification of TKM, called recurrent SOM, that overcomes this limitation. In recurrent SOM, a 'leaked difference vector' is associated to each unit i. It is defined as

$$\mathbf{y}_i(t) = (1-\alpha)\mathbf{y}_i(t-1) + \alpha(\mathbf{x}(t) - \mathbf{w}_i) \tag{21}$$

with $0 < \alpha < 1$. The best matching unit k is the unit that minimizes the norm of this leaked difference vector. This vector is used in place of $\mathbf{x}(t)$ in the learning rule

$$\Delta\mathbf{w}_i = \gamma h_{ik}(\mathbf{y}_i(t) - \mathbf{w}_i) \tag{22}$$

In our comparison, we tested TKM, recurrent SOM, and recursive SOM. In addition, we included recursive neural gas, a model derived from the neural gas algorithm by Martinetz and Schulten (1991) network. Basically, recursive neural gas extends neural gas in the same way as recursive SOM extends SOM. More precisely, the recursive neural gas learning rules are

$$\Delta\mathbf{w}_i^{\mathbf{x}} = \gamma \exp(-\lambda r_i)\left(\mathbf{x}(t) - \mathbf{w}_i^{\mathbf{x}}\right) \tag{23}$$

$$\Delta\mathbf{w}_i^{\mathbf{y}} = \gamma \exp(-\lambda r_i)\left(\mathbf{y}(t-1) - \mathbf{w}_i^{\mathbf{y}}\right) \tag{24}$$

where r_i denotes the *rank* of unit i, that is, the number of units j so that $E_j < E_i$. Parameter λ may be decreased during learning. The equations describing neural activities in recursive neural gas are the same as for recursive SOM. We introduce this algorithm because it achieves better performance than recursive SOM, since topological constraints are avoided. It also demonstrates that our approach has some generality, since other algorithms than the standard SOM can be extended recursively.

In order to compare these networks, we used the binary sequence from the two-states automaton described in the previous section. This non-i.i.d. series has simple temporal statistics, which allowed us to find an optimal quantizer analytically. We used this optimal quantizer as a reference. The maps used for the comparison were two-dimensional grids of size 10×10 units. For each map, parameters were tuned in order to optimize performance. The leak constant of the recurrent SOM was $\alpha = 0.6$. For the recursive SOM, we used $\alpha = 5$, $\beta = 0.5$, and the size of the neighborhood was exponentially decreased during learning, by increasing σ from 0.3 to 10. For recursive neural gas, we used $\alpha = 2$, $\beta = 0.2$, and λ was exponentially decreased during learning, between 0.1 and 0.3. Maps were trained for 3×10^6 iterations, using a constant learning rate equal to 0.1.

Receptive fields and depth were computed using the previous definitions. Results are presented in Fig. 4. Receptive fields are represented as the branches of binary trees. The analytically computed optimal quantizer is represented by the binary tree of Fig. 4D. Binary trees corresponding to the learned representations are shown in Fig. 4A–C.

The depth of a representation by TKM is 1.4 (not

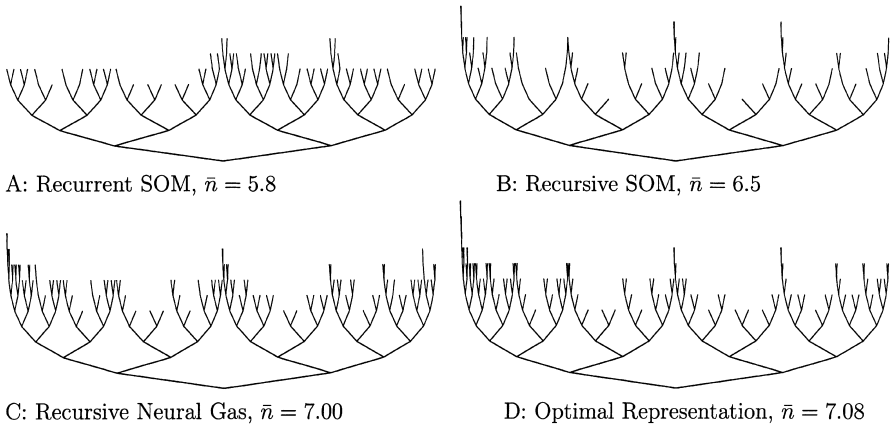

Fig. 4. Binary trees corresponding to the representations generated by recurrent SOM (A), recursive SOM (B), recursive neural gas (C) and to an optimal representation (D). All representations use 100 units. Left branches correspond to '0' and right branches to '1'. Trees (A)–(C) have less leafs than the optimal tree (D), because some units are never selected, or have the same receptive field as other units. Units of recursive SOM and recursive neural gas develop receptive fields whose depth is adapted to the statistics of the data, resulting in trees (B) and (C) that approach the optimal tree (D).

represented in the figure). The representation by recurrent SOM has a depth of 5.8. Fig. 4A shows that the units of recurrent SOM are able to represent long sequences of bits. However, if we compare the shape of trees in Fig. 4A and D, we see that the receptive fields of recurrent SOM are not adapted to the temporal statistics of the input.

In contrast, recursive SOM and recursive neural gas capture the probabilistic structure of the binary sequence; branches are deeper for contexts that have higher probabilities of occurrence, resulting in binary trees (Fig. 4B and C) that resemble the optimal tree (Fig. 4D). Representations by recursive SOM and recursive neural gas have higher depth (6.5 and 7.0, respectively). Note that depth is a log-scaled index of how well units are exploited; for a binary tree, increasing depth by one unit requires 2 times more units. Therefore the differences observed here are high. The difference between trees in Fig. 4B and C is due to topology constraints which limit the performance of recursive SOM. In contrast, the depth of the representation by recursive neural gas (7.0) almost equals the optimum (7.08).

In order to explain the adaptation of the receptive fields in our models to the temporal statistics of the input, we demonstrate a property that characterizes context quantizers of maximal depth:

Theorem 1. *A context quantizer described by a binary tree* $H_1H_2\cdots H_N$ *is optimal if and only if for all* $(i,j,k) \in \{1,\ldots,N\}$, *if* H_i *and* H_j *have the same parent, then* $p_i + p_j \geq p_k$.

Proof. See Appendix A.

Intuitively, Theorem 1 states that depth is maximized when all code-book letters tend to have equal probalilities of being selected. More precisely, a quantizer is optimal if it is not possible to find another quantizer closer to equiprobability. Note that this is consistent with classical vector quantization results.

In fact, neurons will have equal probabilities of being selected if the weight vectors $(\mathbf{w}_i^{\mathbf{x}}, \mathbf{w}_i^{\mathbf{y}})_{1\leq i\leq N}$ have a point density proportional to the joint density of $(\mathbf{x}(t), \mathbf{y}(t-1))$. Martinetz, Berkovich, and Schulten (1993) have demonstrated that the point density of neural gas tends to be proportional to the input density to the power $D/(D+2)$, where D is the dimension of the input vector. In the case of recursive neural gas $D = \dim(\mathbf{x}) + \dim(\mathbf{y})$, and $\dim(\mathbf{y}) = N$ is high ($N = 100$ in the previous experiment). Therefore we can consider that receptive fields are equiprobable, which explains the near-optimal performance of recursive neural gas. Point density is more difficult to analyze for the original SOM (Cottrell, 1997; Ritter & Schulten, 1998). However, these considerations qualitatively explain why recursive SOM learns receptive fields that are adapted to the temporal statistics of the input.

6. Quantization of a continuous series

In the previous sections we considered discrete inputs, and a discrete quantity, the length of correct reconstruction, was averaged in the computation of depth. In this section, we want to test recursive SOM on a continuous series. Depth, however, cannot be defined for continuous inputs, because the distance between an input vector and its prototype is continuous. Hence, we need to adapt our measure of performance.

For continuous inputs, we define the receptive field of a neuron as the mean input sequence that triggers its selection. Note that we need to restrict receptive fields to a time window. However, this time window is defined for monitoring only, and its size does not affect the function of our model. We still consider that receptive fields are the letters of a code-book used for representing context. The

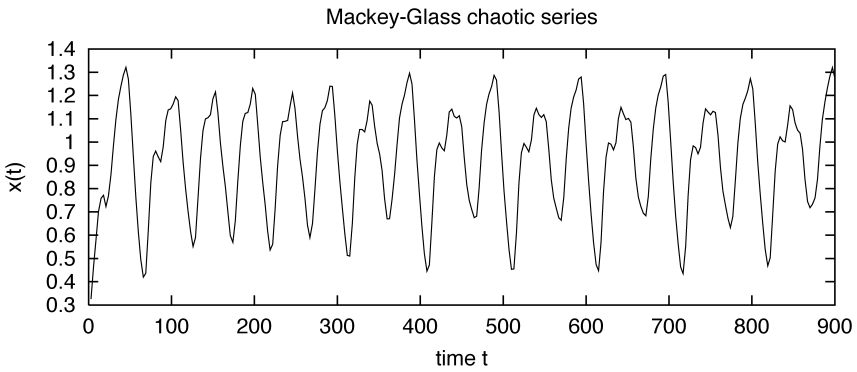

Fig. 5. The Mackey–Glass chaotic time series was used as input to the network.

quantization error of a neuron is defined as the standard deviation associated to its receptive field, and it is a vector of corresponding length. A general quantization error, used as an overall performance measure of the map, is defined as the expected quantization error of the selected unit.

As an input, we used the Mackey–Glass time-series (Mackey & Glass, 1977), a well-known dynamic system defined by the differential equation

$$\frac{\mathrm{d}x}{\mathrm{d}\tau} = bx(\tau) + \frac{ax(\tau - d)}{1 + x(\tau - d)^{10}} \tag{25}$$

For $d > 16.8$, the series $x(\tau)$ is chaotic and highly sensitive to initial conditions. In this paper, we used the parameter values $a = 0.2$, $b = -0.1$ and $d = 17$. A sequence of values of $x(\tau)$ is plotted in Fig. 5, starting from random initial conditions.

The Mackey–Glass series has immediate regularities; that is, past or future values of the series are partially predictable from its current value, with no information about other events. Hence, in order to evaluate how temporal regularities are learned by our model, it is necessary to assess the importance of these immediate regularities. This can be done by comparison to a model that has no sensitivity to time. For this reason, we compared recursive SOM to the original SOM, which is memoryless. We also added recurrent SOM to the comparison. All maps were of size 10×10 and were trained for 150 000 iterations. For recursive SOM two sets of parameters were tested, $(\alpha = 2, \beta = 0.06)$, and $(\alpha = 2, \beta = 0.02)$. The recurrent SOM used a leak constant $\alpha = 0.1$. The input was the Mackey–Glass series sampled every three time units, that is, $x(t)$ for $t = \tau/3$.

The receptive fields of SOM and recursive SOM ($\beta = 0.06$) after training are displayed in Fig. 6. Black lines represent the preferred input sequence of each unit. Gray intervals correspond to confidence intervals of size plus and minus one standard deviation. This standard deviation is the quantization error.

For the SOM, quantization error is minimal for units 0 and 99, which correspond to extreme values of the input. This is not surprising, since it is easy to predict that the series will decrease once it has reached its maximal value, and will in turn increase once it has reached its minimum. In contrast, the receptive fields corresponding to intermediate values of the input have a much higher standard deviation. This is because the SOM has no sensitivity to time. In contrast, the receptive fields of recursive SOM are more structured; their diversity reflects the temporal profiles observed in the time series. Topological organization can be observed on the map. For most units of the recursive SOM, the standard deviation of past events is low; this demonstrates that temporal information is retained, and that the responses of units are highly specific.

Quantization errors are plotted in Fig. 7 for SOM, recurrent SOM and recursive SOM. The time window was of size 30. Since the SOM cannot represent temporal information, it provides a reference to which performance of other models can be compared. The SOM develops a very accurate representation of the present event, with a quantization error close to zero. The error curve then oscillates, which reflects temporal regularities of the series. The recurrent SOM demonstrates the same overall level of error as SOM. However, oscillations are phase-shifted. Experimentally, we observe that the leak constant of the recurrent SOM controls the value of this phase shift. This shift indicates that temporal information is represented in recurrent SOM. However, the representation of certain parts of the signal (e.g. at $t - 3$) is performed at the expenses of other parts of the signal. Hence, the overall performance of recurrent SOM is not significantly better than for SOM. In contrast, the recursive SOM demonstrates a significantly reduced quantization error over time. The sole exception is the current input, for which the SOM makes a much smaller error. This is because the resources of recursive SOM are used for representing past events, while SOM can use all its neurons for representing the current input with high accuracy. For recursive SOM, oscillations of the error curves have a reduced amplitude, which reflects its adaptation to the temporal regularities of the input.

The influence of β can be observed in the difference between the curves for $\beta = 0.06$ and 0.02. For recent events, error is smaller with $\beta = 0.02$ and higher with 0.06. For events older than eight time steps it is the contrary. Hence, parameter β controls the trade-off between accuracy

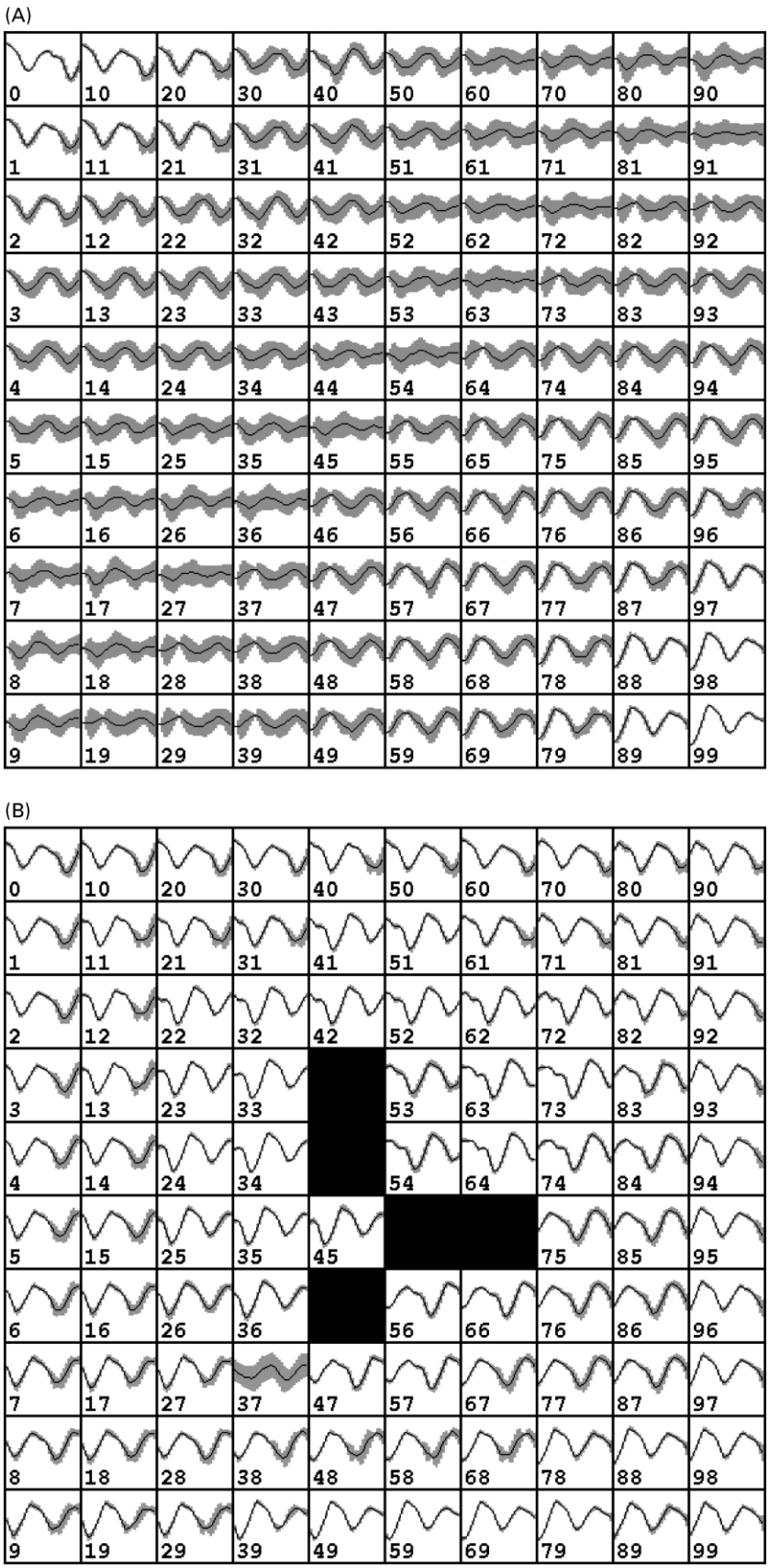

Fig. 6. Comparison of the temporal receptive fields of the units of SOM (A) and recursive SOM (B). Maps were trained on the Mackey–Glass time series. The preferred input sequence of each unit is in black, with the most recent input on the left. The gray intervals reflect the standard deviation, or quantization error. Receptive fields are displayed for 30 past events. In the recursive SOM, five units are never selected. This is due to the difficulty to find a low-dimensional mapping for temporal sequences, which are represented by high dimensional vectors.

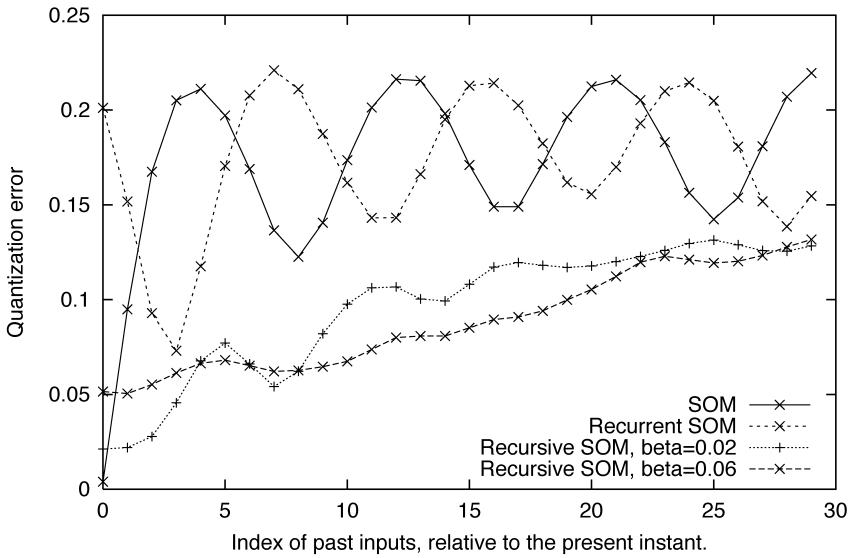

Fig. 7. Temporal quantization error of different models trained on the Mackey–Glass series. This error is defined as the mean difference between the input sequence and the receptive field of the selected unit. Quantization error reflects the uncertainty about past events. Thirty past inputs are considered here.

and depth of the representation. Accuracy was not relevant for discrete inputs, because reconstruction was either correct or incorrect. For continuous inputs, however, representing vectors with higher accuracy has a cost, it must be performed at the expenses of depth.

7. Conclusions

We have demonstrated that local representations of temporal sequences can efficiently be learned by a modified SOM that uses time-delayed feedback. The design of recursive SOM is homogeneous and straight-forward. Stability, which has been the main issue for this type of network, is achieved by choosing an appropriate transfer function.

Recursive SOM successfully generalizes several key features of SOM to time. Self-organization occurs based on the temporal statistics of the input, and topological organization of the map is based on temporal order. Vector quantization properties are generalized to sequences, as reflected by the adaptation of the receptive fields to the temporal structure of the input. In the discrete case, this adaptation is reflected by the depth of the receptive fields; branches of the corresponding trees are higher where the temporal dependencies are stronger. As a result, representations are nearly optimal, in the sense that the network's memory of past events is maximized. Although we did not investigate optimality in the continuous case, adaptation of the receptive fields probably leads to the same type of results for continuous inputs.

The choice of a transfer function is a crucial question, that would deserve further investigation. The choice we have made was empirical, motivated by the need to enforce stability. However, in addition to its effect on stability, the transfer function has an influence on performance, as demonstrated in Section 4, when α and β were too high. We observed that a poor information transfer results in poor performance. Hence, a less empirical approach to the choice of a transfer function would be to maximize information transfer while imposing stability. This could be achieved by adapting the transfer function to the joint density of $(\mathbf{x}(t), \mathbf{y}(t-1))$; the same type of adaptation has been successful in other applications of the Infomax principle (Bell & Sejnowski, 1995; Nadal & Parga, 1994).

Acknowledgments

The author would like to thank Peter Dominey, Lewis Bott and two anonymous reviewers for their helpful comments.

Appendix A. Proof of Theorem 1

We use the terms 'children, descendants' to designate nodes that are higher than the current node of a tree, and 'parents, ancestors' to designate nodes that are lower. Hence, the root of a tree is the ancestor of all nodes.

A context quantizer where two nodes are equal, say $H_1 = H_2$, is not optimal, since a better quantizer $\{H_1' H_2' H_3 H_4 \cdots H_N\}$ can be built by replacing H_1, H_2 with their children H_1' and H_2'. Therefore, if a quantizer $\{H_1 H_2 \cdots H_N\}$ maximizes $\bar{n}$, then all its nodes are distinct.

Suppose now that $\{H_1 H_2 \cdots H_N\}$ maximizes $\bar{n}$. Consider two leafs H_2 and H_3 that have the same parent, and another leaf H_1 at another position in the tree. If we prune the tree at H_2 and H_3, as shown in Fig. A.1, and use the remaining

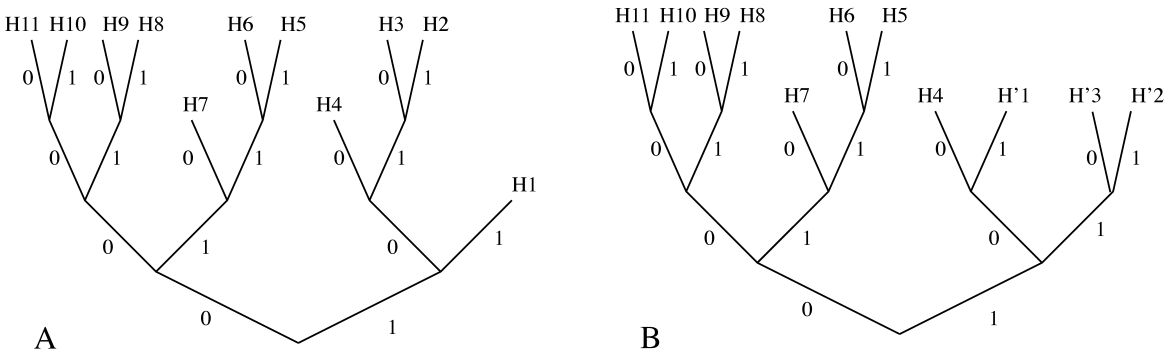

Fig. A.1. Transformation of a binary tree: (A) initial tree; (B) transformed tree. Two neighboring leafs, H_2 and H_3 are cut, and the tree is extended at leaf H_1.

letter to extend the tree at node H_1, we build a new quantizer $\{H'_1H'_2H'_3H_4\cdots H_N\}$, where H'_1 is the parent of H_2, H_3, and H'_2 and H'_3 are the children of H_1. The depth $\bar{n}$ of $\{H_1H_2\cdots H_N\}$ is:

$$\bar{n} = p_1n_1 + p_2n_2 + p_3n_3 + \sum_{i=4}^{N} p_in_i \tag{A1}$$

where n_i is the depth of node H_i, and here $n_2 = n_3$. With the same notations, the depth of $\{H'_1H'_2H'_3H_4\cdots H_N\}$ is:

$$\bar{n}' = p'_1n'_1 + p'_2n'_2 + p'_3n'_3 + \sum_{i=4}^{N} p_in_i \tag{A2}$$

Since H'_1 is the parent of H_2 and H_3, $p'_1 = p_2 + p_3$ and $n'_1 = n_2 - 1 = n_3 - 1$. Similarly, $p'_2 + p'_3 = p_1$ and $n'_2 = n'_3 = n_1 + 1$. Hence, the difference $\bar{n}' - \bar{n}$ is equal to $p_1 - p_2 - p_3$. In addition, $\bar{n}' \leq \bar{n}$ because $H_1H_2\cdots H_N$ maximizes $\bar{n}$. This implies that $p_1 \leq p_2 + p_3$. Therefore the condition of Theorem 1 is necessary.

Now assume that two distinct quantizers $\{H_1H_2\cdots H_N\}$ and $\{H'_1H'_2\cdots H'_N\}$ satisfy the condition of Theorem 1, and that the depth $\bar{n}$ of $H_1H_2\cdots H_N$ is maximal. The depth of $\{H'_1H'_2\cdots H'_N\}$ is denoted by $\bar{n}'$. We can choose a node u in the first quantizer that is not in the second one:

$$\exists H_u \in \{H_1H_2\cdots H_N\}, \qquad H_u \notin \{H'_1H'_2\cdots H'_N\} \tag{A3}$$

Similarly:

$$\exists H_{u'} \in \{H'_1H'_2\cdots H'_N\}, \qquad H_{u'} \notin \{H_1H_2\cdots H_N\} \tag{A4}$$

In addition, H_u can be chosen so that no node of $\{H'_1H'_2\cdots H'_N\}$ is a descendant of H_u. We may even put an additional constraint on the choice of H_u, which is that the brother H_v of H_u is in $\{H_1H_2\cdots H_N\}$. This is because if H_v does not belong to $\{H_1H_2\cdots H_N\}$, then one can choose another H_u among the children of H_v, and repeat this until both H_u and H_v belong to $\{H_1H_2\cdots H_N\}$. Similarly, one can choose two brother nodes $H_{u'}$ and $H_{v'}$ in $\{H'_1H'_2\cdots H'_N\}$, that are not in $\{H_1H_2\cdots H_N\}$, and so that there is no node of $\{H_1H_2\cdots H_N\}$ above $H_{u'}$ and $H_{v'}$.

Since both trees have the same root, there exists a unique ancestor $H_{w'}$ of H_u and H_v that is in $\{H'_1H'_2\cdots H'_N\}$. Similarly, $H_{u'}$ and $H_{v'}$ have a unique ancestor H_w in $\{H_1H_2\cdots H_N\}$. By hypothesis, $p_u + p_v \geq p_w$ and $p_{u'} + p_{v'} \geq p_{w'}$. Since $H_{w'}$ is a parent of H_u and H_v, we have $p_{w'} \geq p_u + p_v$. Similarly, $p_w \geq p_{u'} + p_{v'}$. Combining the previous inequalities yields:

$$p_u + p_v = p_{w'} = p_w = p_{u'} + p_{v'} \tag{A5}$$

This means that w' is the parent of u and v, and w is the parent of u' and v' (code-book letters must have probabilities greater than zero since $\{H_1H_2\cdots H_N\}$ is optimal). Replacing H'_u, H'_v, H'_w by H_u, H_v, H_w in $\{H'_1H'_2\cdots H'_N\}$ does not change the depth. We may repeat the above operation a finite number of times, after which both quantizers will be equal, showing that $\bar{n} = \bar{n}'$.

References

Bell, A., & Sejnowski, T. (1995). An information maximisation approach to blind separation and blind deconvolution. *Neural Computation*, *7*(6), 1129–1159.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166.

Briscoe, G., & Caelli, T. (1997). Learning temporal sequences in recurrent self-organising neural nets. In A. Sattar (Ed.), Advanced topics in artificial intelligence (pp. 427–435). *Proceedings of the 10th Australian Joint Conference on Artificial Intelligence, AI'97*, Berlin: Springer.

Chappell, G. J., & Taylor, J. G. (1993). The temporal Kohonen map. *Neural Networks*, *6*, 441–445.

Cottrell, M. (1997). Theoretical aspects of the SOM algorithm. *Proceedings of the Workshop on Self-Organizing Maps'97* (pp. 246–267). Helsinki University of Technology, Espoo, Finland.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*, 179–211.

Euliano, N., & Principe, J. (1996). Spatio-temporal self-organizing feature maps. *Proceedings of the International Conference on Neural Networks*, 1900–1905.

Hoekstra, A., & Drossaers, M. (1993). An extended Kohonen feature map for sentence recognition. *Proceedings of the International Conference on Artificial Neural Networks*, 404–407.

Huffman, D. A. (1952). A method for the construction of minimum redundancy codes. *Proceedings of the Institute of Radio Engineers*, *40*(9), 1098–1101.

James, D. L., & Miikkulainen, R. (1995). SARDNET: A self-organizing feature map for sequences. *Advances in Neural Information Processing Systems*, *7*, 577–584.

Kangas, J. (1994). *On the analysis of pattern sequences by self-organizing maps*. PhD Thesis, Helsinki University of Technology.

Kohonen, T. (1989). Self-organization and associative memory (3rd ed). Berlin: Springer.

Kohonen, T. (1997). Self-organizing maps (2nd ed). Berlin: Springer.

Kopecz, K. (1995). Unsupervised learning of sequences on maps with lateral connectivity. *Proceedings of the International Conference on Artificial Neural Networks*, 431–436.

Koskela, T., Varsta, M., Heikkonen, J., & Kaski, K. (1998). Time series prediction using recurrent SOM with local linear models. *International Journal of Knowledge-based Intelligent Engineering System*, *2*(1), 60–68.

Lang, K. J., Waibel, A. H., & Hinton, G. E. (1990). A time-delay neural network architecture for isolated word recognition. *Neural Networks*, *3*, 23–43.

Mackey, M. C., & Glass, L. (1977). Oscillations and chaos in physiological control systems. *Science*, *197*, 287–289.

Martinetz, T., Berkovich, S., & Schulten, K. (1993). Neural-gas network for vector quantization and its application to time-series prediction. *IEEE Transactions on Neural Networks*, *4*(4), 558–569.

Martinetz, T., & Schulten, K. (1991). A neural-gas network learns topologies. In T. Kohonen, K. Mäkisara, O. Simula, & J. Kangas (Eds.), (pp. 397–402). *Proceedings of the International Conference on Artificial Neural Networks*, Amsterdam, Netherlands.

Mozayyani, N., Alanou, V., Dreyfus, J., & Vaucher, G. (1995). A spatio-temporal data coding applied to Kohonen maps. *Proceedings of the International Conference on Artificial Neural Networks*, 75–79.

Nadal, J., & Parga, N. (1994). Non linear neurons in the low noise limit: A factorial code maximizes information transfer. *Network*, *5*, 565–581.

Privitera, C. M., & Morasso, P. (1994). The analysis of continuous temporal sequences by a map of sequential leaky integrators. *Proceedings of International Conference on Neural Networks*, 3127–3130.

Ritter, H., & Schulten, K. (1988). Convergence properties of Kohonen's topology conserving maps: Fluctuations, stability, and dimension selection. *Biological Cybernetics*, *60*, 59–71.

Scholtes, J. C. (1991). *Kohonen feature maps in natural language processing*. Technical Report CL-1991-01, Institute for Language, Logic and Information, University of Amsterdam.

Varsta, M., Heikkonen, J., & Millan, J. D. R. (1997). Context-learning with the self-organizing map. *Proceedings of the Workshop on Self-Organizing Maps'97* (pp. 197–202) Espoo, Finland: Helsinki University of Technology.

Vesanto, J. (1997). Using the SOM and local models in time-series prediction. *Proceedings of the Workshop on Self-Organizing Maps'97* (pp. 209–214) Espoo, Finland: Helsinki University of Technology.