Modelling Memory Functions with Recurrent Neural Networks Consisting of Input Compensation Units: I. Static Situations

Simone Kühn[1*], Wolf-Jürgen Beyn[2] and Holk Cruse[1]

[1] *Department of Biological Cybernetics, Faculty of Biology, University of Bielefeld, Bielefeld, 33501, Germany*

[2] *Department of Mathematics, University of Bielefeld, Bielefeld, 33501, Germany*

* Correspondence to: Simone Kühn
Mail: Simone.Kuehn@uni-bielefeld.de
Phone: +49 521/ 106 5533
Fax: +49 521/ 89010

Abstract

Humans are able to form internal representations of the information they process – a capability which enables them to perform many different memory tasks. Therefore, the neural system has to learn somehow to represent aspects of the environmental situation; this process is assumed to be based on synaptic changes. The situations to be represented are various as for example different types of static patterns but also dynamic scenes. How are neural networks consisting of mutually connected neurons capable of performing such tasks?

Here we propose a new neuronal structure for artificial neurons. This structure allows to disentangle the dynamics of the recurrent connectivity from the dynamics induced by synaptic changes due to the learning processes. The error signal is computed locally within the individual neuron. Thus, online learning is possible without any additional structures. Recurrent neural networks equipped with these computational units cope with different memory tasks. Examples illustrate how information is extracted from environmental situations comprising fixed patterns to produce sustained activity and to deal with simple algebraic relations.

1 Introduction

From early childhood on humans brains have a fundamental ability: they build up representations. Brains and their constituents, the neurons, are specialised to represent aspects of the environment which means that these neurons or groups of neurons "stand for" those aspects. This information coded within neural circuits can be multifaceted. Information of objects like a tree or a chair can as well be represented as rules, for example underlying grammar in language, or dynamic events like the movement of one person toward another. To start with we want to focus on the two first examples: We propose a new neuronal architecture that is able to deal with these problems. Its ability to represent dynamic situations is treated in a companion paper.

This paper contributes to a larger project the goal of which is to develop a complete memory system containing a large number of memory engrams, able to learn, behave and reason using this memory structure. This memory system will consist of two parts. One represents a collection of many situation models, the other some kind of "Selector net" that is able to connect sensory input with the appropriate situation model. Although this Selector net, of course, comprises an important element of the complete system, we will not deal with this problem in this paper, but concentrate on the question how individual situation models may be constructed and how they can be learned when a situation is given via sensory input. Situation models stand for information stored in LTM, but also for information stored in STM or a working memory (Baddeley, 1986, 1992). Such a working memory allows us to hold representations of external information actively in memory, at least for a short time, to be able to act within and react to the world. In various experiments the properties of working memory have been investigated applying so-called delayed response tasks. The pioneer work has been done by Fuster and Niki (Fuster and Alexander, 1971; Fuster, 1973; Niki, 1974a, 1974b). In continuing this work many studies using electrophysiological recordings show a stimulus-specific, enhanced delay activity in several brain areas (for reviews see Fuster, 1995; Miyashita and Hayashi, 2000; Wang, 2001). This sustained internal activity in the absence of the external stimulus is argued to be the neural substrate of working memory. An exciting example concerning the representation of dynamic scenes is given by Umiltà et al. (2001). So called mirror neurons are shown to be active when a grasping movement to an object is observed, even when the object is hidden behind an occluder. However, the neuron remains silent when the same movement is observed but the monkey knows that there is no object hidden behind the occluder.

Another important capability human brains have is representing rules. This becomes apparent when regarding language learning. Marcus et al. (1999) have shown that statistical learning mechanisms – which are, of course, not called into question to exist – do not exhaust the child's repertoire of learning mechanism. They performed experiments showing that already 7 month old babies are able to extract simple algebraic relations from acoustic input. The babies were able to distinguish between three word sentences consisting of made-up words and following either the condition "ABA" or "ABB". As the test words were totally new and the sentences were the same length the babies could not distinguish them based on transitional probabilities or statistical properties.

Representing such algebraic relations means representing "open-ended abstract relationships for which we can substitute arbitrary items. For instance, we can substitute any value of x into the equation $y = x + 2$." (Marcus et al., 1999; see also Chomsky, 1980; Pinker and Prince, 1988; Pinker, 1991; Marcus et al., 1995; Marcus, 2001). The point made in the study is that humans, including young babies, are not only capable of generalising due to statistical learning mechanisms in order to build correct sentences as described, but are capable of representing the underlying general rule.

However, even insects can cope with similar problems. Honey bees can solve "delayed matching to sample" tasks which allowed Giurfa et al. (2001) to show that honey bees are able to learn the concept of "symmetry". These authors could further show that honey bees are able to learn the concept of "difference" or "oddity" which implied these insects to be able to cope with "delayed non-matching to sample" tasks. Drosophila is assumed to be able to construct a dynamic representation of an optical pattern that has disappeared behind an occluder and expects it to appear again at the other side of that occluder (Strauss and Pichler 1998).
For many of the different abilities of brains computational models have been proposed. The most promising among them are models with recurrently coupled neurons because they seem to resemble natural neuronal assemblies best. This approach will also be followed in this paper. As the tasks mentioned require an internal representation of the current external situation, some form of learning is necessary. To model example-based learning different forms of error backpropagation (Rumelhart et al., 1986; Hertz et al., 1991) are widely used training procedures for both feed-forward and recurrent neural networks (RNNs). But backpropagation is often considered to be biologically implausible because the error signal has to be provided externally and a specific additional network is required that is able to propagate these error signals.
Additionally, most artificial recurrent networks exposed to learning situations suffer from two severe problems. On the one hand, training is particularly difficult in RNNs because two different dynamics are intertwined: There is the dynamics of the RNN itself, the properties of which depend on distribution and size of the weights. If, on the other hand, these weights are changed additionally due to the learning procedure, a second dynamic process is introduced that interacts with the first one. Therefore, neural and synaptic dynamics are coupled in a very intricate way (Del Giudice et al., 2003) making the control of the network a hard problem (Steil, 1999). This difficulty is often solved by application of off-line training procedures, that separate the dynamics of the network from the dynamics of the training procedure like in Contrastive Hebbian Learning (Movellan, 1990, Baldi and Pineda, 1991; Xie and Seung, 2003) or training echo state networks (Jaeger and Haas, 2004), or by hand-tuning the parameters (e.g. Seung et al., 2000). But neither a cut-off of the feedback loop nor hand-tuning seems to be biologically plausible. Online learning algorithms like real-time recurrent learning (e.g. Williams and Zipser, 1989b), in contrast, are often very slow and computationally very expensive concerning storage capacity and computation time (see (Williams and Zipser, 1989b; Schmidhuber, 1992; Doya, 1995). Furthermore, they are non-local and would require a large additional network structure when being applied to biological systems.

In this paper, we propose a new biologically inspired computational circuit consisting of neuronal units called *Input Compensation Unit* (IC Unit). Using these units allows to disconnect the dynamics of the recurrent network from the dynamics due to the learning procedure. Therefore, these units allow for an easy training of RNNs in an online mode to model the two tasks mentioned above – i.e. holding an item in memory which means learning the representation of static patterns, and representing simple algebraic relations. Additionally it is possible that a network equipped with those units is also able to learn dynamic situations. This is described in the companion paper (Kühn and Cruse, 2006).
The circuit acts within a neuronal unit and incorporates a learning rule that formally corresponds to the delta rule (Widrow and Hoff, 1960), but does not require a separate network for backpropagating the error. Each neuron only needs local information directly available via its synaptic connections. The error is determined within each neuron. Therefore, the training procedure is unsupervised as no global trainer is necessary and each neuron relies on local information only. Consequently, the computational costs are very low. Thus, our

model overcomes the main objections against traditional approaches in training recurrent neural networks. A very similar rule has been proposed by (Kalveram, 2000) for training feedforward networks. The difference to our approach is discussed below (Section 5).
The final goal behind this approach is to design a memory system that contains the representation of many different situations. Such situations may comprise static or moving objects or describe connections between a sensory input and a motor output, analogue to so-called motor primitives as proposed by Wolpert and Kawato (1998), for example. In some contrast to these motor primitives, the situation models proposed here are not separable in forward models and inverse models, but each situation model combines both aspects in one holistic model (see also Cruse 2003). The units of these RNN models - like those of the somewhat more complex MMC nets (Steinkühler and Cruse 1998) - may be interpreted as showing properties of mirror neurons, or canonical neurons (Gallese and Lakoff 2005), because no functional separation is possible between neurons being related to perceptual or to motor tasks. These units correspond to what Gallese and Lakoff (2005) termed "multimodal neurons".
The view, that different situations are stored by specific networks, is supported by physiological findings (Fogassi et al., 2005). Studying mirror neurons, i.e. neurons which likewise represent sensory as well as motor aspects, Fogassi and colleagues (2005) have shown that different neurons are activated when identical movements are either observed or performed which however belong to a different context (e.g. eating or placing). As mentioned, in this paper we do not deal with the question of how cooperation or selection of different situation models may be organised, but first concentrate on the basic structure of such situation models.
The situation models proposed here consist of very simple recurrent neural networks which may, therefore, also be suited to serve as models for cognitive properties of animals like insects. Thus we follow the general idea of Beer (2003) who proposes to study "minimal cognition" first and then try to develop this system to be able to cope with higher cognitive functions. An excellent example for this approach is given by Feldman and Narayanan (2004) and Narayanan (1999).
In the following (Section 2) we want to specify the tasks in more detail the network should be able to deal with. The structure of the circuit proposed is described in Section 3. After having presented the results (Section 4) the paper concludes with a discussion of the networks' properties including some biological interpretations.

2 The tasks

2.1 Learning a static pattern to produce sustained activity

The first task the network should cope with is to represent a fixed static pattern consisting of analogue values that is given as input to produce sustained activity even if the input pattern disappears. Specifically, the task is as follows: The recurrent network consists of at least n units. As an example a network for $n = 3$ is depicted in Figure 1a. Any n-dimensional input vector is provided to the network. The learning algorithm should change the weights in a way that all units of the network adopt activations that correspond to the input and maintain their activation even after the external input is switched off.
Which values should the weights take if a fixed input vector is presented? Assume that we have a network with n units with output values $x_1, x_2, \ldots x_n$ and the input vector consists of the components $\mathbf{a} = (a_1, a_2, ..., a_n)^{\mathrm{T}}$. The task is then to find a weight matrix $\mathbf{W}$ with $\mathbf{a} = \mathbf{W} \cdot \mathbf{a}$. This means that the weights of the recurrent network should form a matrix that has the vector $(a_1, a_2, ..., a_n)^{\mathrm{T}}$ as an eigenvector corresponding to the eigenvalue $\lambda = 1$, while all other

eigenvalues satisfy $|\lambda| < 1$. As we have n^2 weights there is a manifold of matrices that fulfil this condition, as n equations determine n degrees of freedom. Therefore, $(n^2 - n)$ of the n^2 weights can be chosen arbitrarily. For $n = 3$ one possible solution is given by matrix W1:

$$1/(a_1 + a_2 + a_3) \cdot \begin{pmatrix} a_1 & a_1 & a_1 \\ a_2 & a_2 & a_2 \\ a_3 & a_3 & a_3 \end{pmatrix} \qquad \text{(W1)}$$

With $\mathbf{1} = (1,1,1)^T$, W1 can be rewritten as $[1/(\mathbf{1}^T \cdot \mathbf{a})] \cdot \mathbf{a} \cdot \mathbf{1}^T$. W1 is a skew projector. It projects onto $span\{\mathbf{a}\}$ along the space that is orthogonal to $\mathbf{1}$. Such a network does not only stabilise an input situation given by vector $(a_1, a_2, a_3)^T$, but any multiple of this vector. If the initial activations of the units are set to values that deviate from this condition, the network relaxes to a vector that obeys this relation, i.e., to a multiple of $(a_1, a_2, a_3)^T$. The network can therefore be described as forming an attractor consisting of a two-dimensional subspace that is described by the plane $a_1 x_1 + a_2 x_2 + a_3 x_3 = 0$. This network is only neutrally stable. Neutral stability means that if any weight is changed arbitrarily, the activations of the units increase to infinity or may decrease to zero. Therefore, a learning mechanism is needed that automatically stabilises the weights against disturbances as for example disturbances due to synaptic noise.

2.2 *Representing simple algebraic relations*

As a further task, the network should be able to store simple algebraic relations. Here, we deal with two examples of such relations: First, the results obtained by Marcus et al. (1999) should be simulated with the network proposed here. Marcus and colleagues found, that the infants tested were able to extract abstract algebra-like rules that represent the relationship between variables such as "the first item X is the same as the third item Y". Simulations of two experiments have to be performed: In the first one the network has to be trained with external input of structure "ABA" and in the second one with external input of structure "ABB". The network can be tested afterwards (just like the babies) with consistent input, i.e. input resembling the structure of the training phase, or with inconsistent input. The test input has to consist of variables not yet presented during the training phase to prevent learning based on transitional probabilities. The babies in the experiments described above paid attention to the inconsistent sentences for a longer period of time (for details see Marcus et al., 1999).

The second task to be learnt by the network is more general by nature: It should be able to represent simple linear equations. The network should be able to sum up two variables, i.e. to represent all possible configurations of x_1 and x_2 that result in a value $x_3 = x_1 + x_2$. If we do not wish to apply a 3D look-up table for all possible cases, the mechanism, i.e. the underlying rule or equation, should be represented which can then be applied to any given values. For this specific example, an easy solution is to use two input units x_1 and x_2, the output of which is fed in as input to a third unit, with weights of unity. However, there are two tasks related tightly: The task $x_3 = x_1 + x_2$ also implies that $x_1 = x_3 - x_2$ and $x_2 = x_3 - x_1$. Of course, two further independent nets could be constructed that can solve these additional tasks. This solution would require a kind of selector network that decides which of the three nets should be used depending on the task given.

A simpler solution is to form one "holistic" network that represents the complete situation and can solve all three tasks. This recurrent network is given by the equation $\mathbf{x}(t+1) = \mathbf{W} \cdot \mathbf{x}(t)$ or, for $n = 3$, by:

$$x_1(t+1) = w_{11} \cdot x_1(t) + w_{12} \cdot x_2(t) + w_{13} \cdot x_3(t)$$
$$x_2(t+1) = w_{21} \cdot x_1(t) + w_{22} \cdot x_2(t) + w_{23} \cdot x_3(t)$$
$$x_3(t+1) = w_{31} \cdot x_1(t) + w_{32} \cdot x_2(t) + w_{33} \cdot x_3(t)$$

Here, $\mathbf{x}(t)$ is the vector describing the actual activation of the n units ($n = 3$ in our case) and $\mathbf{x}(t+1)$ the vector describing the activation in the following time step. $\mathbf{W}$ describes the n^2 weights w_{ij} ($i = 1$ to n, $j = 1$ to n). If the weights are chosen appropriately, this system has stable solutions that fulfil the equation $x_1 + x_2 - x_3 = 0$. An appropriate weight matrix is given by matrix W2:

$$\begin{pmatrix} 0 & -1 & 1 \\ -1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \qquad \text{(W2)}$$

The tasks regarded here can be understood as pattern completion tasks: Given any two values as input, after relaxation the network will provide all three values x_1, x_2, and x_3 at the output, i.e., a correct solution in any case. Therefore, depending on the input variables chosen, any of the three subtasks can be solved by this network. A correct solution is even found if only one input value is defined. As this latter task is underconstrained, different solutions are possible. The solution actually chosen by the network depends on its earlier state.

3 The model: A recurrent neural network with IC Units

3.1 Structure of IC Units

In this section we explain the architecture of a network that can cope with both tasks specified above and can, as will be shown in the companion paper, also learn to represent dynamic situations. To explain the structure of the network and to explicate its individual units let us consider a network that consists of n recurrently connected units. An example of a three-unit network is shown in Figure 1a.

Each individual neuron x_i ($i = 1$ to n) is equipped with a special internal structure (Fig. 1b) described in the following. The dendritic tree is partitioned into two regions: One region with fixed synapses, whose presynaptic neurons belong to sensory neurons transmitting the external input a_i. To simplify matters each neuron can only be stimulated by one external stimulus. As the synaptic weight is fixed it is not specified in Figure 1b and 1c. The second dendritic region is characterised by active synapses w_{ij}, whose presynaptic neurons are components of the recurrent network (Fig. 1a) and are recurrently connected to neuron x_i. *Active synapses* are synapses which can be either potentiated or depressed (Montgomery and Madison, 2004) and thus are exposed to learning. Therefore, the activation of a single neuron is determined by an external component a_i and an internal component, the weighted sum of the internal recurrent inputs s_i. The weighted sum of the internal recurrent inputs of neuron x_i is given by

$s_i(t) = \sum_{j=1}^{n} w_{ij}(t) \cdot x_j(t)$ or, for the complete network, $\mathbf{s}(t) = \mathbf{W}(t) \cdot \mathbf{x}(t)$.

Such a splitting in an external and a recurrent component can also be found in the model described by Del Giudice et al. (2003). Please recall that *s* describes the internal input, not the output of the unit.

[insert Figure 1 about here]

3.2 Training the synaptic weights

The overall goal in both tasks mentioned above is to represent the external situation **a** (a static pattern or several examples following an algebraic relation) perceived via the sense organs within the network. 'Representing the external situation' can be defined as follows: If the weighted sum of the internal recurrent inputs s_i of neuron x_i equals the external input a_i, this stimulus is represented within the network because then the external input is no longer needed to elicit the activation characterising the stimulus a_i. In order to reach this goal the synaptic weights w_{ij} have to be adapted in a learning process.

As has been mentioned above, a major problem with training RNNs is that the dynamics of the network are superimposed by the dynamics due to the learning procedure. Both dynamics could however be separated, if, during training, the overall output x_i would always equal the external input (i.e. $x_i = a_i$) independent of the actual learning state, i.e., independent of the actual values of the weights w_{ij}. This can be achieved if we determine the output x_i by

$$x_i(t+1) = a_i(t) = s_i(t) + a_i(t) - s_i(t) = s_i(t) + \delta_i(t) \qquad (1)$$

with $\delta_i(t) = a_i(t) - s_i(t)$. The corresponding circuit is shown in Fig. 1b (solid lines).

To attain the overall goal, the weights w_{ij} have to be changed such that $x_i(t+1) = s_i(t)$ or, in other terms, $\delta_i(t) = 0$. This can be obtained by application of the learning algorithm

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij} \text{ with } \Delta w_{ij} = \varepsilon \cdot x_j(t) \cdot \delta_i(t) \qquad (2)$$

with $\varepsilon > 0$ being the learning rate (for more detailed information about the choice of ε see Appendix). This learning algorithm formally corresponds to the delta rule. However, in contrast to the traditional approach, the delta error is here assumed to be determined and propagated locally within each neuron (Fig. 1b, dashed arrows) as has been proposed by Kalveram (2000) for feedforward networks or Jaeger and Haas (2004) for echo state networks. Application of the rule depicted in equation (2) leads to a weight change until $\delta_i(t) = 0$, i.e., until the sum s_i of the weighted recurrent inputs equals the external input a_i. We call units with this internal structure *Input Compensation Units* (IC Units), because this circuit compensates the effect of the external input, independent of the actual state of the recurrent weights.

To be able to address this memory content later, it is necessary to prevent the network to automatically adapt to each new input situation. Thus, once the synaptic connections have learnt the specific input situation, further learning is stopped. A simple solution is to finish learning after the error δ_i has fallen below a given threshold because then external situation is represented within the network. To simplify matters, in the simulations shown here further learning is stopped, if the summed squared error $E(t) = \sum_{i=1}^{n} \delta_i(t)^2$ of the entire network has fallen below a given threshold.

3.3 Extension of the neuronal structure

To account for working memory capabilities, it should also be possible to sustain the activation once induced by a stimulus. As explained above the overall output of an IC Unit as shown in Figure 1b will, however, decay to zero after the external stimulus vanishes. This is due to the property of the IC Units, that the output always equals the input. Thus, the network cannot remain active to act as working memory.

In order to be able to sustain the activation, the architecture requires an extension. If a_i is smaller than s_i in a unit shown in Fig. 1b the output activation x_i decreases, because then negative δ-values (recall that $\delta_i(t) = a_i(t) - s_i(t)$) are added to s_i. This effect can, however, be avoided if we rewrite equation (1) by using rectifiers, which means that only the positive part of the function is transmitted. The rectifier is marked by a $+$ in the following equations.

For an explanation, we will first consider only positive input values ($a_i(t) \geq 0$). If the weights are small at the beginning of training, for example zero, which means that $s_i(t_0) = 0$, we can assume that during training the condition $0 \leq s_i(t) \leq a_i(t)$ is fulfilled which is biologically plausible. With this assumption, the condition $a_i(t) \geq 0$ can be replaced by $s_i(t) \geq 0$ and equation (1) can be rewritten:

$$x_i(t+1) = s_i(t) + [a_i(t) - s_i(t)]_+, \qquad \text{for } s_i(t) \geq 0 \tag{3.1}$$

Following (3.1), x_i still corresponds to s_i, even if $a_i(t) < s_i(t)$. Therefore, using this rectifier, the external input can indeed be switched off after training is finished, i.e. $a_i(t) = 0$, and no changes occur to the output (if training has not yet been finished completely, the activation of the units will slowly decrease to zero, see Discussion). Note, that the rectifiers do not influence the δ-value used for learning.

Furthermore, we can generalise this condition for negative input values ($a_i(t) \leq 0$): If we again assume that the weights are small at the beginning of learning, for example zero, we can state $0 \leq |s_i(t)| \leq |a_i(t)|$, because during learning s_i will approach a_i starting from zero also for negative input values a_i. Correspondingly, we can now replace the condition $a_i(t) \leq 0$ by $s_i(t) \leq 0$. This leads to the second equation

$$x_i(t+1) = s_i(t) - [-a_i(t) + s_i(t)]_+, \qquad \text{for } s_i(t) \leq 0 \tag{3.2}$$

Both equations (3.1) and (3.2) are depicted in the circuit diagram in Fig. 1c. The condition $s_i(t) \geq 0$ and $s_i(t) \leq 0$ are represented by the clipping functions. The two rectifiers used in equations (3.1) and (3.2) are depicted in the lower part of the circuit (Fig. 1c). This circuit fulfils three requirements:

(i) It allows to apply both positive and negative input values a_i.

(ii) After training is finished, it maintains its activation after the external input has been switched off.

(iii) It shows the same training properties as the linear version (Fig. 1b), if the condition $0 \leq |s_i(t)| \leq |a_i(t)|$ is fulfilled.

The results shown in the following were obtained by using this expanded network. Note that the nonlinear expansions applied are only necessary for being able to use the network after learning is finished, i.e. in the testing mode. The learning procedure as such can still be

described by a linear approach. As before and during training the activations of the neurons are only determined by the external input values a_i due to their *input compensation* property, the dynamics resulting from the weight changes do not affect the dynamics of the complete network and therefore do not cause stability problems.

4 Results

4.1 Learning a static pattern to produce sustained activity

Training the network. Let us first consider the case of a network consisting of three units that receives an external, fixed input vector $(a_1, a_2, a_3)^{\mathrm{T}}$. Numerical investigations reveal the following results which can also be proven to hold generally (see Appendix in Chapter 3.6).

If all nine weights including the diagonal weights, by which each neuron influences itself directly, are allowed to be learnt and all weights are set to zero at the beginning, the IC learning procedure (Fig. 1, Eq. (2)) provides the solution shown by matrix W3

$$1/\left(a_1^{\,2} + a_2^{\,2} + a_3^{\,2}\right) \cdot \begin{pmatrix} a_1a_1 & a_1a_2 & a_1a_3 \\ a_2a_1 & a_2a_2 & a_2a_3 \\ a_3a_1 & a_3a_2 & a_3a_3 \end{pmatrix} = \left(1/\left(\mathbf{a}^{\mathrm{T}}\mathbf{a}\right)\right) \cdot \mathbf{a}\mathbf{a}^{\mathrm{T}} \qquad \text{(W3)}$$

Matrix W3 is the orthogonal projector onto $span\{\mathbf{a}\}$. In geometrical terms, the behaviour of an individual unit k can be described as follows: Assume the network consists of n units and is trained with a vector $\mathbf{a}$. The output of unit k is determined by

$x_k(t+1) = w_{k1}x_1(t) + w_{k2}x_2(t) + \ldots + w_{kn}x_n(t)$,

which describes a linear function in an $(n+1)$-dimensional space. This function corresponds to an n-dimensional hyperplane that contains the origin and, after training, the $(n+1)$-dimensional vector $(a_1, a_2, \ldots a_n, a'_k)^{\mathrm{T}}$. a'_k and x'_k describe the additional dimension given by the output value $x_k(t+1)$. This hyperplane also contains the $(n-1)$-dimensional subspace that is contained in the n-dimensional space (x_1 to x_n). This subspace is orthogonal to vector $(a_1, a_2, \ldots a_n)^{\mathrm{T}}$. In other words, this hyperplane could be constructed in the following way: The hyperplane defined by $x'_k = 0$ is rotated around the vector orthogonal to $(a_1, a_2, \ldots a_n)^{\mathrm{T}}$ until it contains the vector $(a_1, a_2, \ldots a_n, a'_k)^{\mathrm{T}}$. For $n = 2$ and $k = 2$, this process is schematised in Figure 2.

[insert Figure 2 about here]

The network adopts solution W4 (for a proof see Appendix), if during training all diagonal weights are constantly set to zero:

$$\begin{pmatrix} 0 & a_1a_2/\left(a_2^{\,2} + a_3^{\,2}\right) & a_1a_3/\left(a_2^{\,2} + a_3^{\,2}\right) \\ a_1a_2/\left(a_1^{\,2} + a_3^{\,2}\right) & 0 & a_2a_3/\left(a_1^{\,2} + a_3^{\,2}\right) \\ a_1a_3/\left(a_1^{\,2} + a_2^{\,2}\right) & a_2a_3/\left(a_1^{\,2} + a_2^{\,2}\right) & 0 \end{pmatrix} \qquad \text{(W4)}$$

In general, matrix W4 is asymmetric. The *n*-dimensional hyperplane described by unit *k* contains the origin and the vector $(a_1, a_2, ...a_n, a_k)^T$, but now contains the k^{th} coordinate axis instead of the vector orthogonal to $(a_1, a_2, ...a_n)^T$ as was the case for (W3).

Solution (W4) is of practical interest, because starting from this solution, a manifold of solutions can be constructed by replacing the diagonal weights by arbitrary positive values d_i first and then normalising all weights of unit *i* by multiplication with $1/(1+d_i)$. Parameters d_i can be interpreted as damping factors: The larger d_i, the slower the network approaches to a stable solution. A special treatment of the diagonal weights is plausible in biological systems, because these weights correspond to the only synapses by which the neurons are connected to themselves.

Addressing the memory content. After having trained the network with a certain input vector **a** this external input can be switched off without changing the output; thus, due to the internal connections built up during learning the network keeps the activity induced by the external stimuli even if the stimuli are no longer present.
How does the network react to incorrect input? If for a limited period of time an input vector is provided to the network that does not correspond to its stored vector, the network relaxes to a stable state that corresponds to its stored vector or a multiple thereof, after having switched off the input. Therefore, the network has the ability of pattern completion. For a network characterised by matrix W3, the stable state is reached immediately. For matrix W4 the relaxation takes some time depending upon value d_i ($d_i > 0$). A given ε-neighbourhood of the stable state is reached the faster, the more similar input vector and stored vector (or its nearest multiple) are.

4.2 Representing simple algebraic relations

Training the network. The second task addressed in the Introduction and Section 2 was to learn algebraic rules, as given in the condition ABA or ABB on the one hand and equations like $x_3 = x_1 + x_2$ on the other hand. Such tasks require that not only one vector is learnt, but a solution for all vectors is found that fulfil the respective condition.
Providing a network consisting of IC Units with input vectors following the former condition ABA (e.g. $(5,1,5)$, $(2,3,2)$) leads to weight matrix (W5):

$$\begin{pmatrix} 0.5 & 0 & 0.5 \\ 0 & 1 & 0 \\ 0.5 & 0 & 0.5 \end{pmatrix} \qquad \text{(W5).}$$

Training the network with the second condition applied by (Marcus et al., 1999), namely ABB (e.g. $(5,1,1)$, $(2,3,3)$) another weight matrix is obtained:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0.5 & 0.5 \end{pmatrix} \qquad \text{(W6).}$$

The solution of this task allows a simple geometrical interpretation. In a 3D-space (x_1,x_2,x_3) the solution of the task (A,B,B) is given by a 2D-plane with $x_2 = x_3$ and any value of x_1, i.e. a

plane that is perpendicular to the $x_1 - x_2$ plane. Matrix (W6) corresponds to this solution with $d_2 = d_3 = 1$, and d_1 being infinite ($d_1 / (d_1 + 1) = 1$).

The second task mentioned in Section 2.2 requires a solution for all vectors fulfilling the equation $\sum_{i=1}^{n} c_i x_i = 0$ for given $c_i \in \mathbb{R}$, i.e. all vectors of an $(n-1)$-dimensional hyperplane containing the origin. Geometrically, for $n = 3$, the solution is given by a plane in the 3D coordinate system that contains all points given by the coordinates that fulfil the equation $c_1 x_1 + c_2 x_2 + c_3 x_3 = 0$. Therefore, the solution is completely defined if three points are given. As $(0,0,0)$ is already a solution, only two further examples (not collinear with $(0,0,0)$) are sufficient to specify the solution. Generally, the solution for any task described by $\sum_{i=1}^{n} c_i x_i = 0$ with fixed coefficients $c_i \in \mathbb{R}$ is uniquely defined if $n-1$ examples are presented to the network that form an $(n-1)$-dimensional subspace and do not contain the origin.

Application of a network with IC Units actually leads to this solution. To illustrate this ability, we again use a three unit network. The task to be trained is $x_3 = x_1 + x_2$. Any two training examples fulfilling the equation could be used (e.g. $(5,-1,4)$, $(-1,4,3)$). The same solution (W7) is obtained whether the training examples are presented in periodic epochs or in random order:

$$\begin{pmatrix} 2/3 & -1/3 & 1/3 \\ -1/3 & 2/3 & 1/3 \\ 1/3 & 1/3 & 2/3 \end{pmatrix} \qquad \text{(W7)}$$

Here, all nine weights were allowed to learn. Matrix W7 can be interpreted to be a special case of matrix W2 that is expanded by application of a damping factor $d_i = 2$ as explained above. If, however, the diagonal weights are constrained and always set to zero, we obtain a solution that corresponds to matrix W2 (see section 2). These results are based on numerical investigations; a general proof is still pending.

Addressing the memory content. If a network trained on either of the two conditions ABA or ABB is provided with a consistent input (e.g. $(7,1,7)$ for the first condition ABA), it immediately stabilises at this values even if the values have not been presented to the network before, i.e. are totally new. If, in turn, inconsistent input is presented to the network (e.g. $(5,1,1)$ for the first condition ABA), the activation of the unit not matching the condition asymptotically approaches the correct value.

To address the memory content after having trained all the weights according to the task of representing the summation (or, based on matrix W4, after the application of any positive damping factors), the network is provided with a vector **a** the first component a_1 and the second component a_2 of which are fixed to certain values while the third a_3 is set to zero. In the end, the third component should be the sum of the other components.

In each case, the network provides a solution that fulfils $x_3 = x_1 + x_2$. But it is not necessarily the case that $x_1 = a_1$ and $x_2 = a_2$. This condition is fulfilled in two cases:

(*i*) $x_1 = a_1$ and $x_2 = a_2$, if $|a_1| > |a_2|$ and $|a_2| \geq \frac{|a_1|}{2}$

(*ii*) $x_1 = a_1$ and $x_2 = a_2$, if $|a_2| > |a_1|$ and $|a_1| \geq \frac{|a_2|}{2}$.

If, however, $|a_1| > |a_2|$ and $|a_2| < \frac{|a_1|}{2}$, we obtain $x_1 = a_1$ and $x_2 = a_2 - \frac{a_1 + a_2}{2}$; and if $|a_2| > |a_1|$ and $|a_1| \geq \frac{|a_2|}{2}$, we obtain $x_1 = a_1 - \frac{a_1 + a_2}{2}$ and $x_2 = a_2$.

Therefore, if all $a_i \in (i) \vee (ii)$ unit x_3 approaches asymptotically the value $x_3 = a_1 + a_2$. Nevertheless, in the other cases as mentioned the network still stabilises at a value x_3 following the summation task $x_3 = x_1 + x_2$. Thus, the trained network is able to cope also with this pattern completion task. Correspondingly, solving the equation for the other variables is possible, too.

5 Discussion

The work presented here forms an essential step towards a broader goal, that is to develop a complete memory system which can be used for learning information, for recognition, and for retrieving stored information to perform actions, but may also be used for mental simulation. Basic elements of such a complete memory system are situation models that are able to represent the relevant information to serve a working memory and/or a long term memory. In this section, we discuss the properties and the biological plausibility of the neuronal units (IC units) introduced here, we then discuss the properties of the complete networks consisting of such IC units and compare them with related approaches. Finally, we discuss how these networks can be used for short term (working) memory and long term memory functions.

In this paper we propose *Input Compensation Units* (IC Units) as a new internal structure for artificial neurons that can be used as a basic building block of recurrent neural networks and allows for an efficient training of the synaptic weights. RNNs consisting of these IC Units and being trained in the described way have two main advantages over traditional approaches in training recurrent neural networks making them biologically more realistic:

First, the learning algorithm can be applied online, i.e. without cutting the recurrent connections, because the learning dynamics are disentangled from the dynamics of the recurrent network as such. Thus, the individual units behave as if belonging to a feed-forward net. This is possible due to the following properties: As the sum of the weighted internal inputs is subtracted from the external input, the output of the neuron always equals the size of the external input and is therefore independent of its learning state (Eq. 3.1 and 3.2). In other words, as the built-in compensation mechanism always replaces that part of the input signal that corresponds to the sum of the recurrent signals, the global dynamics of the network is protected from the learning dynamics. Therefore, no stability problems arise here due to weight changes. During the training procedure, the weights stabilise at values guaranteeing that in the end the summed recurrent input s_i equals the external input a_i. After learning is completed, and the summed internal input equals the external input, the latter can be switched off without changing the activation of the network.

Second, the synaptic weights of each neuron are adapted using local information only. The single neuron does not rely on information about the activation of whole network but only to information directly available at its synaptic connections just like real neurons. Consequently, the computational costs are very low – in contrast to many other training procedures (e.g. Williams and Zipser, 1989a; Schmidhuber, 1992) as no specific network for determination of the error and for its backpropagation is needed.

As mentioned in Sect. 2.1, many solutions are possible. One, however trivial, solution for all static cases is given by the identity matrix. However, the identity matrix never showed up as a result except for such cases in which this matrix was the only solution, i.e. when several input vectors were given which did not fulfil the basic equation.

5.1 Biological plausibility

To implement the mechanism described the neuron has to distinguish between external input and input supplied by the recurrent connections of the network. How is this possible in a biological network? It is known (e.g. Kandel et al., 2000) that different types of synapses exist; the strength of one type does not easily change whereas other synapses show variation depending on activity. Additionally, physiological findings show, that the dendritic tree of a neuron is subdivided into different computational subunits for chemical signals such as changes in concentration of ions or other second messengers; this compartmentalisation is considered to be the basis of local modifications of the dendritic properties to achieve, for example, input-specific changes of synaptic weights (Helmchen, 1999) and it is also important from a computational perspective (Mel, 1999). Therefore, a different treatment of sensory input to the neuron and the recurrent internal input might well be possible.
Furthermore, some speculations concerning potential molecular mechanisms underlying the internal structure of the IC Units are possible; basic building blocks necessary to realise the algorithm proposed here can be found in real neurons (e.g. Kandel et al., 2000): Several pathways are known that increase and others that decrease the concentration of substances that influence the insertion of AMPA receptors in the synaptic membrane, for example. It is widely assumed, that the kinetics and magnitude of NMDA receptor mediated Ca^{2+} signal determine the sign of synaptic modification (Kirkwood et al., 1993; Cummings et al., 1996). A large increase of Ca^{2+} favours the activation of kinases which results in a phosphorylation of AMPA receptors; a lower increase in contrast favours the activation of phosphatases which results in a dephosphorylation of AMPA receptors (e.g. Lisman, 1989; Cormier et al., 2001).

5.2 Capabilities of the network

Being able to scale the properties of a network with its size is crucially important for a model in order to serve as a biologically plausible brain model. The architecture of many models has to be additionally constrained to scale it by, for example, restricting the connectivity to local neighbourhoods only (Sejnowski et al., 1988).

The model we used, the building blocks of which are IC Units, does not suffer from scaling problems as long as the learning rate ε is chosen small enough according to $0 < \varepsilon < \frac{2}{\mathbf{a}^{\mathrm{T}} \cdot \mathbf{a}}$ (see Appendix) in the case of training the network to represent static situations. Thus, the more units the network has, the smaller the learning rate has to be in order to obtain stable solutions. Therefore, this IC model seems to be promising for further applications and it is possible to train more realistic networks consisting of a large number of neurons.

Representing static patterns. Using these units it is possible to solve several memory tasks. First, static input patterns can be applied; due to the built-in learning mechanism the weights adapt in a way that the activations of the units remain fixed even after the external input signal has been switched off, thus producing sustained activity in the network.
It has been suggested to use attractor dynamics of coupled neurons provided with strong feedback for modelling these states of enhanced activity (Wilson and Cowan, 1973; Amari, 1977; Hopfield, 1982; Zipser et al., 1993; Amit, 1995; Kühn and Cruse, 2005; for reviews on neurocomputational models see Durstewitz et al., 2000; Del Giudice et al., 2003). However, the performance of many of the proposed models is highly dependent on fine tuning the network parameters such as synaptic strength. If parameters only deviate slightly from the tuned values, the networks tend to diverge (Wang, 2001). In contrast, our model does not require fine-tuning of the weights as it automatically adapts to the current input situation.

When providing the network with a vector different from the stored one, the stored vector or a multiple of it is reproduced. This property can be interpreted as an error correction mechanism (or the capability to generalise) as it has been described for Hopfield nets (Hopfield, 1984; for a more detailed comparison with other recurrent neural networks see below).
Additionally, if a part of the vector is not specified by the input, i.e. a component of the input vector is set to zero, the network shows the ability of pattern completion: It finds an appropriate activation for the unspecified units.

Representing algebraic relations. There has been a heated debate on the claim made by Marcus et al. (1999) that it is not possible to replicate their results with simple recurrent neural networks (see Seidenberg and Elman, 1999). The problem with connectionist-like models is that they are not able to generalise the abstract patterns to new words and are thus dependent on the input choice. They cannot abstract the underlying rule as it is necessary for the task described in the Introduction and in Section 2. The model presented here does not represent any word explicitly but only the rule of an open-ended abstract relationship, in this case a simple algebraic relation. If the network is provided with consistent input it immediately stabilises on these activation values, whereas it needs some time to relax on the inconsistent condition. This matches with the results of the experiments performed by Marcus et al. (1999). The time the network needs to relax when provided with inconsistent input can be interpreted as to correspond to the longer time of attention the infants paid to sentences being inconsistent with the trained ones in the experiments carried out by Marcus et al. (1999). Therefore it is possible to simulate the experimental results obtained by Marcus et al. (1999) with networks consisting of IC Units.
Similarly, such algebraic rules may also underlie other grammatical phenomena as for example building English sentences with plural agreement from an arbitrary set of noun and verb phrases. In this sense humans know for example that a correct English sentence can be formed by combining any plural noun phrase with any verb phrase with plural agreement: From the two phrases “Bart and Lisa”, which is a plural noun phrase, and “played in the garden”, which is a verb phrase with plural agreement, we can infer that “Bart and Lisa played in the garden” is a correct English sentence. Here as well, networks that rather represent the abstract relations between the items than the single words may underlie the ability to build correct sentences.

The network can also be trained to represent any linear task $\sum_{i=1}^{n} c_i x_i = 0$ when only some (at least $n-1$) correct training examples are presented. The network forms a holistic representation of this algebraic relation implying the capability of pattern completion also in this task: If $n-1$ variables are given, the remaining variable is determined by the network. If, during recall, fewer variables are given and the task is therefore underdetermined, the network still provides a correct solution. The task is not solved by using a look-up table, but by representing the underlying mechanism.

The tasks described in Section 4.2 are characterised by homogeneous equations $\sum_{i=1}^{n} c_i x_i = 0$. However, this network can also be applied to tasks that require non homogeneous equations $\sum_{i=1}^{n} c_i x_i = h_i$ with constant values h_i. This corresponds to the introduction of a ‘bias unit’ often used in neural networks. The network can simply be extended by such a bias unit by adding a unit, which is assumed to have a constant activation of 1. The weight of this bias unit corresponds to the value h_i and can be trained using the same algorithm explained above.

Human's internal representations are not necessarily static by nature. As already mentioned by Johnson-Laird (1983) internal representations could be dynamic, i.e. they show time-dependent behaviour. This claim is underpinned by recall experiments showing that memory can be influenced by the observed movement (e.g. direction and speed) of an object (Freyd and Finke, 1984). Such dynamical systems can also be modelled by a network consisting of Input Compensation Units as will be explained in a companion paper (Kühn and Cruse, 2006).

5.3 Comparison with other recurrent neural networks

The underlying idea of the Input Compensation Units corresponds to the clamped phase in Contrastive Learning (CL) procedures (Movellan, 1990; Baldi and Pineda, 1991). The advantages of CL are the possibility to train networks with hidden units on the one hand and to use nonlinear activation functions on the other hand. Up to now it has not been tested how the IC approach could deal with nonlinearities and hidden units. These are certainly the next problems to be tackled. Preliminary investigations are encouraging.
But there are three main differences between the two approaches: First, in all examples of the CL approach the weights of the feedback connections are assumed to be symmetric with the feedforward connections. In networks consisting of IC Units the weights are not constraint.
Second, in contrast to CL only one phase is applied and no oscillations between a phase with a teacher signal and one without a teacher signal are necessary.
Third, in CL the dynamics of the network are separated from the dynamics due to the learning procedure by definition as the dynamical equations are first run until convergence to a fixed point and then the weights are updated (Xie and Seung, 2003). In doing so, the problem of intertwining two interacting dynamics does not arise. But it is biologically not plausible that the synapses only then change, after the dynamics of the network has settled. For biological systems this "waxing and waning" of the synapses is assumed to not be explicitly uncoupled from the network's activity but on the contrary explicitly *dependent* on the network's activity. The latter is the case in the neuronal units presented here: the updating of the weights is performed online, i.e. in each single time step and there is no necessity to decouple it explicitly from the network. Therefore, the IC approach appears to be nearer to biological reality.
The training procedure used here is based on the principle of teacher forcing (e.g. Williams and Zipser, 1989a; Doya, 1995; Jaeger and Haas, 2004): the actual output of a unit is replaced by the teacher signal in the subsequent computation. This principle permits online learning and has been applied by other approaches like real-time recurrent learning for RNNs (e.g. Williams and Zipser, 1989b). The problem with real-time recurrent learning is that it is computationally very intensive concerning storage and time and – moreover – the algorithm is non-local because each weight needs the knowledge of the complete recurrent weight matrix and the error vector. RNNs consisting of IC Units are trained using local information only and therefore the computational costs are very low.
To alleviate the problem of computational costs, a number of approaches have been put forward like, for example, the modification of the real-time recurrent learning algorithm by Schmidhuber (1992) which reduces at least the computational time but still needs quite large storage capacities.
Kalveram (2000) also proposed a learning algorithm formally corresponding to the delta rule like the IC approach incorporated on the level of the individual neuron. This has been applied to feedforward networks. The weights of external inputs are trained by providing the unit with the desired output. This input corresponds to the fixed external input used here but has to be

switched off after training. In contrast our networks comprise memory units that are activated via the external input (see also below).
Other examples trying to reduce computational costs are the echo-state networks (Jaeger and Haas, 2004) and, quite similar besides using spiking neurons, the liquid-state machines (Maass et al., 2002). These types of networks need more units to equip the reservoir but are able to learn complex dynamic behaviour. Storing static patterns has not been addressed within these approaches. It will be shown in a companion paper that learning dynamic patterns is also possible with RNNs consisting of IC Units (Kühn and Cruse, 2006).

Similarities could be figured out, too, between the IC networks and Hopfield (Hopfield, 1982; 1984) networks on the one hand and MSBE networks (Cruse, 2006, and below) on the other hand. What is the difference between the weight matrices resulting from the training procedure presented here to that of those other types of recurrent neural networks? The former are defined by symmetric weights and bounded activation functions. The units used here do not have bounded activation functions. Symmetric weights could, but do not necessarily result from application of the IC algorithm. Symmetric weights arise in matrices W2, W5, W6 and W7, but not in W3 and W4. Therefore, application of IC Units does generally not lead to Hopfield type networks.
MSBE networks are derived in the following way. If an equation with n variables $\sum_{i=1}^{n} v_i \cdot x_i = 0$ is solved for each variable x_i, a set of equations is obtained. If each of these n equations is considered to represent the computation performed by the corresponding neuron i, the network represents *M*ultiple *S*olutions for the *B*asic *E*quation $\sum_{i=1}^{n} v_i \cdot x_i = 0$ and is termed therefore MSBE network. For $n = 3$, for example, the basic equation $v_1 x_1 + v_2 x_2 + v_3 x_3 = 0$ being resolved for x_1, x_2 and x_3 leads to a weight matrix

$$\begin{pmatrix} 0 & v_2/v_1 & v_3/v_1 \\ v_1/v_2 & 0 & v_3/v_2 \\ v_1/v_3 & v_2/v_3 & 0 \end{pmatrix} \qquad \text{(W8)}$$

MSBE networks, like Hopfield networks, can be considered as autoassociators that have the property of pattern completion. Unlike Hopfield networks, that show discrete attractors, the attractor points of MSBE networks form a smooth, bounded space.
The weights follow the condition $w_{ij} \cdot w_{ji} = 1$. So the MSBE network is symmetric only for $v_1 = v_2 = v_3$. As described above for (W4), the weight matrix W8 can be extended by the introduction of damping factors d_1, d_2, and d_3.
Inspection of the different weight matrices obtained by the learning procedure applied to the IC Units reveals that some, but not all matrices fulfil the condition $w_{ij} = 1/w_{ji}$. Matrix W2 fulfils the condition, matrix W3 only when applying a damping factor $d_i = \frac{a_i^2}{a_1^2 + a_2^2 + a_3^2} - 1$ and W4 when applying $d_i = \frac{a_i^2}{a_1^2 + a_2^2 + a_3^2} - 1$. This means that the IC algorithm can but does not necessarily produce weight distributions typical for MSBE networks. The latter is the case in particular, when in contrast to all examples used here, the weights are not all set to zero at the beginning of training.

5.4 Working memory and long term memory functions

In various experiments properties of the working memory have been investigated (Del Giudice et al., 2003). In electrophysiological recordings stimulus-specific, enhanced activity can be observed which is assigned to be a feature of active working memory and enables animals to hold items in memory for some time. If no further attention is applied to the content of memory, it vanishes after a short time.

This property can also be found in our model: After presenting a static stimulus the activation of the artificial neurons is enhanced. During learning the weights approach the final values characterising the neutrally stable state only asymptotically. Therefore, in more natural situations, training is finished with non-ideal weight values. Hence, after an input has been presented to the network and later switched off, the activation of the network does not remain constant, but decreases to zero with a velocity depending on how closely the ideal values have been approximated during training (note that the weights themselves maintain their values). This property may be considered as corresponding to the function of working memory, the content of which disappears if no specific attention is applied to maintain this content for a longer time. The velocity of this decrease of activation depends on the quality of learning, i.e., on learning time.

At the same time, the network can be considered to represent a passive memory (Fuster, 1995). If, after an activated network has been returned to zero activation, the input a_1, a_2, a_3 is presented again later, it would immediately activate the network.

As described above the weight values are only changed by means of the learning algorithm (Eq. 2), i.e., only when an external input is given. However, weights may also decay spontaneously (as do synapses), but with a long time constant (e.g. hours or days). Under this condition, the IC Units alone were not sufficient to explain long term memory. The following additional mechanism could, however, be applied: If the excitation has been strong enough, or has been repeated sufficiently often, a special mechanism may come into action that prohibits synaptic decay and weights may stay fixed. In other words, the network forms a long term memory only after this fixation process has been performed (for a review of observations concerning switches between discrete states of synapses see Montgomery and Madison, 2004). In contrast to the architecture explained above, this additional mechanism would imply that not every input is maintained in the long term memory. Rather the system would be able to select frequent or salient information, and only such information is stored permanently.

6 Appendix: Learning a static pattern to produce sustained activity

6.1 Proof of convergence – training all the weights

During the training phase the $n \times n$ weight matrix $\mathbf{W}(t)$ is updated according to (2) as follows

(A1) $\mathbf{W}(t+1) = \mathbf{W}(t) + \varepsilon \cdot \delta(t) \cdot \mathbf{a}^{\mathrm{T}} = \mathbf{W}(t) + \varepsilon \cdot [\mathbf{I} - \mathbf{W}(t)] \cdot \mathbf{a} \cdot \mathbf{a}^{\mathrm{T}} \quad t = 0,1,2,\ldots$.

We denote by $\mathbf{P}_{\mathrm{a}} = \dfrac{1}{\mathbf{a}^{\mathrm{T}} \cdot \mathbf{a}} \cdot \mathbf{a} \cdot \mathbf{a}^{\mathrm{T}}$ the orthogonal projector onto span$\{\mathbf{a}\}$.

Theorem 1 *Under the assumption*

(A2) $0 < \varepsilon < \dfrac{2}{\mathbf{a}^{\mathrm{T}} \cdot \mathbf{a}}$

the iteration (A1) converges for any $\mathbf{W}(0) = \mathbf{W}_0$ *to the weight matrix*

(A3) $\mathbf{W}_\infty = \mathbf{W}_0 \cdot (\mathbf{I} - \mathbf{P}_{\mathrm{a}}) + \mathbf{P}_{\mathrm{a}}$.

In particular if $\mathbf{W}(0) = 0$ *we obtain* $\mathbf{W}_\infty = \mathbf{P}_{\mathrm{a}}$ *as in* (**W3**).

Proof : We use the following well-known result (Berman and Plemmons, 1994).

Theorem 2 *Let* $\mathbf{X}$ *be a finite dimensional linear space and let* $\mathbf{X}$ *be the direct sum of two of its subspaces* $\mathbf{X}_1$ *and* $\mathbf{X}_2$*, i.e. every* $\mathbf{W} \in \mathbf{X}$ *can be written in a unique way as* $\mathbf{W} = \mathbf{W}_1 + \mathbf{W}_2$ *where* $\mathbf{W}_1 \in \mathbf{X}_1, \mathbf{W}_2 \in \mathbf{X}_2$.

Let $\mathbf{L} : \mathbf{X} \to \mathbf{X}$ *be a linear map such that*

(i) $\mathbf{L} \cdot \mathbf{W} = \mathbf{W}$ *for all* $\mathbf{W} \in \mathbf{X}_1$

(ii) $\mathbf{L}$ *maps* $\mathbf{X}_2$ *into itself and* $|\lambda| < 1$ *for all eigenvalues* λ *of* $\mathbf{L}$ *that belong to eigenvectors in* $\mathbf{X}_2$.

Then the iteration

(A4) $\mathbf{W}(t+1) = \mathbf{L} \cdot \mathbf{W}(t) + \mathbf{R}_2, \ \mathbf{W}(0) = \mathbf{W}_0$

converges for any $\mathbf{W}_0 \in \mathbf{X}$ *and any* $\mathbf{R}_2 \in \mathbf{X}_2$ *to*

$\mathbf{W}_\infty = (\mathbf{W}_0)_1 + \mathbf{W}_2$

where $\mathbf{W}_0 = (\mathbf{W}_0)_1 + (\mathbf{W}_0)_2$ *is the decomposition of* $\mathbf{W}_0$ *and* $\mathbf{W}_2 \in \mathbf{X}_2$ *is the unique solution in* $\mathbf{X}_2$ *of the equation*

(A5) $\mathbf{W}_2 = \mathbf{L} \cdot \mathbf{W}_2 + \mathbf{R}_2$.

We apply this Theorem 1 to (A1) with $\mathbf{X}$ the space of $n \times n$ matrices and

$\mathbf{X}_1 = \{\mathbf{W} \in \mathbf{X} : \mathbf{W} \cdot \mathbf{a} = 0\}, \dim \mathbf{X}_1 = n^2 - n$

$\mathbf{X}_2 = \{\mathbf{b} \cdot \mathbf{a}^{\mathrm{T}} : \mathbf{b} \in \mathrm{R}^n\}, \dim \mathbf{X}_2 = n$.

The decomposition of $\mathbf{W} \in \mathbf{X}$ is given by

$\mathbf{W} = \mathbf{W} \cdot (\mathbf{I} - \mathbf{P}_{\mathrm{a}}) + \mathbf{W} \cdot \mathbf{P}_{\mathrm{a}} = \mathbf{W}_1 + \mathbf{W}_2$

since $\mathbf{W} \cdot (\mathbf{I} - \mathbf{P}_{\mathrm{a}}) \cdot \mathbf{a} = 0$ and $\mathbf{W} \cdot \mathbf{P}_{\mathrm{a}} = \mathbf{b} \cdot \mathbf{a}^{\mathrm{T}}$ with $\mathbf{b} = \dfrac{1}{\mathbf{a}^{\mathrm{T}} \cdot \mathbf{a}} \cdot \mathbf{W} \cdot \mathbf{a}$.

The iteration (A1) has the form (A4) if we define

(A6) $\mathbf{L}\cdot\mathbf{W}=\mathbf{W}\cdot\left(\mathbf{I}-\varepsilon\cdot\mathbf{a}\cdot\mathbf{a}^{\mathrm{T}}\right),\mathbf{R}_2=\varepsilon\cdot\mathbf{a}\cdot\mathbf{a}^{\mathrm{T}}$.

Note that (i) follows from $\mathbf{L}\cdot\mathbf{W}=\mathbf{W}\cdot\left(\mathbf{I}-\varepsilon\cdot\mathbf{a}\cdot\mathbf{a}^{\mathrm{T}}\right)=\mathbf{W}$ for $\mathbf{W}\in\mathbf{X}_1$.

If $\mathbf{W}=\mathbf{b}\cdot\mathbf{a}^{\mathrm{T}}\in\mathbf{X}_2$ then we have

$$\mathbf{L}\cdot\mathbf{W}=\mathbf{b}\cdot\mathbf{a}^{\mathrm{T}}\cdot\left(\mathbf{I}-\varepsilon\cdot\mathbf{a}\cdot\mathbf{a}^{\mathrm{T}}\right)=\mathbf{b}\cdot\left(\mathbf{a}^{\mathrm{T}}-\varepsilon\cdot\left(\mathbf{a}^{\mathrm{T}}\cdot\mathbf{a}\right)\cdot\mathbf{a}^{\mathrm{T}}\right)=\left(1-\varepsilon\cdot\mathbf{a}^{\mathrm{T}}\cdot\mathbf{a}\right)\cdot\mathbf{b}\cdot\mathbf{a}^{\mathrm{T}},$$

therefore $\lambda=1-\varepsilon\cdot\mathbf{a}^{\mathrm{T}}\cdot\mathbf{a}$ is an *n*-fold eigenvalue of $\mathbf{L}$ and $|\lambda|<1$ holds as we have assumed (A2).

Then Theorem 1 is applicable and yields (A3) if we show that (A5) holds for $\mathbf{W}_2=\mathbf{P}_{\mathrm{a}}$.

In fact, $\mathbf{P}_{\mathrm{a}}\in\mathbf{X}_2$ and

$$\mathbf{P}_{\mathrm{a}}-\mathbf{L}\cdot\mathbf{P}_{\mathrm{a}}=\mathbf{P}_{\mathrm{a}}-\mathbf{P}_{\mathrm{a}}\cdot\left(\mathbf{I}-\varepsilon\cdot\mathbf{a}\cdot\mathbf{a}^{\mathrm{T}}\right)=\varepsilon\cdot\mathbf{P}_{\mathrm{a}}\cdot\mathbf{a}\cdot\mathbf{a}^{\mathrm{T}}=\varepsilon\cdot\mathbf{a}\cdot\mathbf{a}^{\mathrm{T}}=\mathbf{R}_2.$$

❒

6.2 Proof of convergence – training with constraints

Now we consider the learning rule (A1) where only certain entries of the weight matrix are updated. We write this as follows:

(A7) $\mathbf{W}(t+1) = \mathbf{W}(t) + \varepsilon \cdot \mathbf{E} \circ (\mathbf{I} - \mathbf{W}(t)) \cdot \mathbf{a} \cdot \mathbf{a}^{\mathrm{T}}$

where E is an $n \times n$ matrix with entries 0 or 1 and where we used the Hadamard product $\mathbf{E} \circ \mathbf{B}$ of $n \times n$-matrices given by

(A8) $(\mathbf{E} \circ \mathbf{B})_{ij} = \mathbf{E}_{ij} \cdot \mathbf{B}_{ij}$.

The entries $\mathbf{W}_{ij}$ with $\mathbf{E}_{ij} = 1$ are updated while those with $\mathbf{E}_{ij} = 0$ are kept constant. In particular, for the choice

(A9) $$\mathbf{E} = \begin{pmatrix} 0 & 1 & \cdots & 1 \\ 1 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 1 & \cdots & 1 & 0 \end{pmatrix}$$

only the weights $\mathbf{W}_{ij}$ with $i \neq j$ are updated.

Theorem 3 *Assume that the matrix* **E** *has no zero row and let* $\varepsilon > 0$ *satisfy for all* $i = 1,...,n$

(A10) $\varepsilon \cdot d_i < 2$, *where* $d_i = \sum_{E_{ij}=1} a_j^{\,2}$.

Then the learning rule (A7) *converges for any* $\mathbf{W}(0) = \mathbf{W}_0$ *to some limit matrix* $\mathbf{W}_\infty$. *In case* $\mathbf{W}_0 = 0$ *the entries of* $\mathbf{W}_\infty$ *are given by*

(A11) $(\mathbf{W}_\infty)_{ij} = \dfrac{a_i \cdot a_j}{d_i}$ *if* $\mathbf{E}_{ij} = 1$

and $(\mathbf{W}_\infty)_{ij} = 0$ *otherwise.*

In particular, for the pattern matrix E from (A9) *with* $n = 3$ *we obtain exactly the matrix* (**W4**). *In case* $\mathbf{E}_{ij} = 1$ *for all i,j we recover the results from Theorem 1.*

Proof: We apply Theorem 1 again with the setting

$$\mathbf{X} = \{\mathbf{W} : \mathbf{E} \circ \mathbf{W} = \mathbf{W}\} = \{\mathbf{W} : \mathbf{W}_{ij} = 0 \text{ if } \mathbf{E}_{ij} = 0\}$$

and

(A12) $\mathbf{L} \cdot \mathbf{W} = \mathbf{W} - \varepsilon \cdot \mathbf{E} \circ [\mathbf{W} \cdot \mathbf{a} \cdot \mathbf{a}^{\mathrm{T}}], \mathbf{R}_2 = \varepsilon \cdot \mathbf{E} \circ [\mathbf{a} \cdot \mathbf{a}^{\mathrm{T}}]$.

The spaces $\mathbf{X}_1$ and $\mathbf{X}_2$ are given by

$$\mathbf{X}_1 = \{\mathbf{W} \in \mathbf{X} : \mathbf{W} \cdot \mathbf{a} = 0\}$$

$$\mathbf{X}_2 = \{\mathbf{W} = \mathbf{E} \circ (\mathbf{b} \cdot \mathbf{a}^{\mathrm{T}}) : \mathbf{b} \in \mathsf{R}^n\}.$$

First note that $\mathbf{L} \cdot \mathbf{W} = \mathbf{W}$ is obvious for $\mathbf{W} \in \mathbf{X}_1$.

In $\mathbf{X}_2$ we choose basis vectors

(A13) $\mathbf{V}_i = \mathbf{E} \circ (\mathbf{e}^i \cdot \mathbf{a}^{\mathrm{T}}),\ i = 1,...,n$

where $\mathbf{e}^i = (0,...,1,...0)$ is the i-th Cartesian basis vector. Note that

$$\left(\mathbf{V}_i \cdot \mathbf{a}\right)_k = \sum \mathbf{E}_{ij} \cdot \left(\mathbf{e}^i\right)_k \cdot \mathbf{a}_j^2 = \left(\mathbf{e}^i\right)_k \cdot \sum_{\mathbf{E}_{ij}=1} \mathbf{a}_j^2 = d_i \left(\mathbf{e}^i\right)_k$$

holds and therefore

(A14) $\mathbf{V}_i \cdot \mathbf{a} = d_i \cdot \mathbf{e}^i$.

Since $\mathbf{E}$ has no zero row we have $d_i > 0$ for all i. Equation (A14) then implies that the vectors $\mathbf{V}_i$ are linearly independent and moreover we find that the vectors $\mathbf{V}_i$ are eigenvectors of $\mathbf{L}$

(A15) $\mathbf{L} \cdot \mathbf{V}_i = \mathbf{V}_i - \varepsilon \cdot d_i \cdot \mathbf{E} \circ \left(\mathbf{e}^i \cdot \mathbf{a}^{\mathrm{T}}\right) = \lambda_i \cdot \mathbf{V}_i,\ \lambda_i = 1 - \varepsilon \cdot d_i$

Condition (A10) guarantees that $|\lambda_i| < 1$ holds for all eigenvalues.

The decomposition $\mathbf{W} = \mathbf{W}_1 + \mathbf{W}_2$, $\mathbf{W}_1 \in \mathbf{X}_1, \mathbf{W}_2 \in \mathbf{X}_2$ is given by

(A16) $\mathbf{W}_2 = \sum_{i=1}^{n} b_i \cdot \mathbf{V}_i,\ b_i = \dfrac{\left(\mathbf{W} \cdot \mathbf{a}\right)_i}{d_i},\ \mathbf{W}_1 := \mathbf{W} - \mathbf{W}_2$.

Note that $\mathbf{W}_1 = \mathbf{W} - \mathbf{W}_2 \in \mathbf{X}$ satisfies by (A14)

$$\mathbf{W}_1 \cdot \mathbf{a} = \mathbf{W} \cdot \mathbf{a} - \sum_{i=1}^{n} b_i \cdot \mathbf{V}_i \cdot \mathbf{a} = \mathbf{W} \cdot \mathbf{a} - \sum_{i=1}^{n} \left(\mathbf{W} \cdot \mathbf{a}\right)_i \cdot \mathbf{e}^i = 0.$$

The decomposition is unique since $\mathbf{W} \in \mathbf{X}_1$ and $\mathbf{W} = \sum_i b_i \cdot \mathbf{V}_i \in \mathbf{X}_2$ implies

$$0 = \mathbf{W} \cdot \mathbf{a} = \sum_{i=1}^{n} b_i \cdot \mathbf{V}_i \cdot \mathbf{a} = \sum_{i=1}^{n} b_i \cdot \mathbf{e}^i = \mathbf{b}.$$

We have now verified the assumptions of Theorem 1.

In order to determine the limit matrix $\mathbf{W}_\infty$ we need to solve (A5) with $\mathbf{R}_2$ given in (A12). The solution is

$$\mathbf{W}_2 = \sum_{i=1}^{n} \frac{a_i}{d_i} \cdot \mathbf{V}_i$$

since by (A15)

$$\mathbf{W}_2 - \mathbf{L} \cdot \mathbf{W}_2 = \sum_{i=1}^{n} \frac{a_i}{d_i} \cdot \left(\mathbf{V}_i - \mathbf{L} \cdot \mathbf{V}_i\right) = \sum_{i=1}^{n} \frac{a_i}{d_i} \cdot \left(1 - \lambda_i\right) \cdot \mathbf{V}_i = \varepsilon \cdot \sum_{i=1}^{n} a_i \cdot \mathbf{V}_i = \mathbf{R}_2.$$

Combining this with (A16) Theorem 1 leads to the limit matrix $\mathbf{W}_\infty$ given for $\mathbf{E}_{ij} = 1$ by

(A17) $\left(\mathbf{W}_\infty\right)_{ij} = \left(\mathbf{W}_0\right)_{ij} + \dfrac{1}{d_i} \cdot \left(a_i - \left(\mathbf{W}_0 \cdot \mathbf{a}\right)_i\right) \cdot a_j$.

In the case $\mathbf{W}_0 = 0$ this leads to formula (A11).

❐

Acknowledgements

This work was funded by the DFG grants GK-518 and the EC-IST project SPARK.

References

Amari S (1977) Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological Cybernetics* 27: 77-87.

Amit DJ (1995) The hebbian paradigm reintegrated: local reverberations as internal representations. *Behavioral and Brain Sciences* 18: 617-626.

Baddeley A (1986) *Working memory*. Oxford: Clarendon Press.

Baddeley A (1992) Working memory. *Science* 255: 556-559.

Baldi P, Pineda F (1991) Contrastive learning and neural oscillator. *Neural Computation* 3: 526-545.

Beer R (2003) The Dynamics of Active Categorical Perception in an Evolved Model Agent. *Adaptive Behaviour* 11: 209-243.

Berman A., Plemmons R.J. (1994), *Nonnegative matrices in the mathematical sciences*. Classics in Applied Mathematics 9, Philadelphia, PA: SIAM Publications.

Chomsky NA (1980) *Rules and Representation*. New York: Columbia University Press.

Cormier RJ, Greenwood AC, Connor JA (2001) Bidirectional Synaptic Plasticity Correlated With the Magnitude of Dendritic Calcium Transients Above a Threshold. *Journal of Neurophysiology* 85: 399-406.

Cruse H (2003) The evolution of cognition – a hypothesis. *Cognitive Science* 27: 135-155.

Cruse, H (2005) Neural Networks as Cybernetic Systems. http://193.30.112.98/brain/archive/cruse [2].

Cummings JA, Mulkey RM, Nicoll RA, Malenka RC (1996) Ca^{2+} signaling requirements for long-term depression in the hippocampus. *Neuron* 16: 825-833.

Del Giudice P, Fusi S, Mattia M (2003) Modelling the formation of working memory with networks of integrate-and-fire neurons connected by plastic synapses. *Journal of Physiology* 97: 659-681.

Doya K (1995) Recurrent networks: Supervised learning. In: *The Handbook of Brain Theory and Neural Networks* (M.Arbib, ed), Cambridge, MA: MIT Press, pp 796-800.

Durstewitz D, Seamans JK, Sejnowski TJ (2000) Neurocomputational models of working memory. *Nature Neuroscience Supplement* 3: 1184-1191.

Feldman J, Narayanan S (2004) Embodied meaning in a neural theory of language. *Brain and Language* 89: 385-392.

Fogassi L, Ferrari PF, Gesierich B, Rozzi S, Chersi F, Rizzolatti G (2005) Parietal lobe: From action organization to intention understanding. *Science* 308: 662-666.

Freyd JJ, Finke RA (1984) Representational momentum. Journal of Experimental Psychology: *Learning, Memory, and Cognition* 10: 126-132.

Fuster JM (1973) Unit activity in the prefrontal cortex during delayed response performance: neuronal correlates of transient memory. *Journal of Neurophysiology* 36: 61-78.

Fuster JM (1995) *Memory in the Cerebral Cortex: An Empirical Approach to Neural networks in the Human and Nonhuman Primate.* Cambridge, MA: MIT Press.

Fuster JM, Alexander GE (1971) Neuron activity related to short-term memory. *Science* 173: 652-654.

Gallese V, Lakoff G (2005) The brain's concepts: the role of the sensory-motor system in conceptual knowledge. *Cognitive Neuropsychology* 22: 455-479.

Giurfa M, Zhang S, Jenett, A, Menzel R, Srinivasan M (2001) The concepts of 'sameness' and 'difference' in an insect. *Nature* 410: 930-933.

Helmchen F (1999) Dendrites as biochemical compartments. In: *Dendrites* (Greg Stuart, Nelson Spruston, Michael Häusser, eds), Oxford: University Press, pp 161-192.

Hertz J, Krogh A, Palmer RG (1991) *Introduction to the theory of neural computation*. Redwood City: Addison-Wesley Pub.

Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the USA* 79: 2554-2558.

Hopfield JJ (1984) Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences of the USA* 81: 3088-3092.

Jaeger H, Haas H (2004) Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication. *Science* 2: 78-80.

Johnson-Laird PN (1983) *Mental models: towards a cognitive science of language, inference, and consciousness*. Cambridge: Cambridge University Press.

Kalveram KT (2000) Sensorimotor sequential learning by a neural network based on redifined Hebbian learning. In: *Perspectives in Neural Computing - Proceedings of the ANNIMAB-1 Conference* (H.Malgrem, M.Borga, L.Niklasson, eds), London: Springer, pp 271-276.

Kandel ER, Schwartz JH, Jessell TM (2000) *Principles of neural science*. New York: McGraw-Hill.

Kirkwood A, Dudek SM, Gold JT, Aizenman CD, Bear MF (1993) Common forms of synaptic plasticity in the hippocampus and neocortex in vitro. *Science* 260: 1518-1521.

Kühn S, Cruse H (2005) Static mental representations in recurrent neural networks for the control of dynamic behavioural sequences. *Connection Science* 17: 343-360.

Kühn S, Cruse H (2006) Modelling memory function with recurrent neural networks consisting of Input Compensation Units II - Dynamic situations. n n.

Lisman JE (1989) A mechanism for the Hebb and the anti-Hebb processes underlying learning and memory. *Proceedings of the National Academy of Sciences of the USA* 86: 9574-9578.

Maass W, Natschläger T, Markram H (2002) Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation* 14: 2531-2560.

Marcus G, Vijayan S, Bandi Rao S, Vishton PM (1999) Rule learning by seven-month-old infants. *Science* 283: 77-80.

Marcus GF, Brinkman U, Clahsen H, Wiese R, Pinker S (1995) German inflection: The exception that proves the rule. *Cognitive Psychology* 29: 189-256.

Marcus GF (2001) *The algebraic mind: Integrating connectionism and cognitive science*. Cambridge, MA: MIT Press.

Mel BW (1999) Why have dendrites? A computational perspective. In: *Dendrites* (Greg Stuart, Nelson Spruston, Michael Häusser, eds), Oxford: University Press, pp 271-289.

Miyashita Y, Hayashi T (2000) Neural representation of visual objects: encoding and top-down activation. *Current Opinion in Neurobiology* 10: 187-194.

Montgomery JM, Madison DV (2004) Discrete synaptic states define a major mechanism of synaptic plasticity. *Trends in Neurosciences* 27: 744-750.

Movellan J (1990) Contrastive Hebbian learning in the continuous Hopfield model. In: *Proceedings of the 1990 Connectionist Models Summer School* (D.Touretzky, J.Elman, T.Sejnowski, G.Hinton, eds), San Mateo, CA: Morgan Kaufmann, pp 10-17.

Narayanan S (1999) Moving right along: a computational model of metaphoric reasoning about events. *Proceed. of the National Conference on Artificial Intelligence* AAAI-99. Orlando Florida (http://www.icsi.berkeley.edu/~snarayan/met.ps

Niki H (1974a) Prefrontal unit activity during delayed alternation in the monkey: I. Relation to direction of response. *Brain Research* 68: 185-196.

Niki H (1974b) Prefrontal unit activity during delayed alternation in the monkey: II. Relation to absolute versus relative direction of response. *Brain Research* 68: 197-204.

Pinker S (1991) Rules of language. *Science* 253: 530-535.

Pinker S, Prince A (1988) On language and connectionism - analysis of a parallel distributed-processing model of language-acquisition. *Cognition* 28: 73-193.

Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by error propagation. In: *Parallel Distributed Processing, volume 1: Foundations* (D.E.Rumelhart, J.L.McClelland, eds.), Cambridge, MA: MIT Press, pp 318-362.

Schmidhuber J (1992) A fixed size storage $O(n^3)$ time complexity learning algorithm for fully recurrent continually running networks. *Neural Computation* 4: 243-248.

Seidenberg MS, Elman JL (1999) Networks are not 'hidden rules'. *Trends in Cognitive Sciences* 3: 288-289.

Sejnowski TJ, Koch C, Churchland PS (1988) Computational Neuroscience. *Science* 241: 1299-1306.

Seung HS, Lee DD, Reis BY, Tank DW (2000) Stability of the memory of eye position in a recurrent network of conductance-based model neurons. *Neuron* 26: 259-271.

Steil JJ (1999) Input-Output Stability of Recurrent Neural Networks. Göttingen: Cuvillier Verlag.

Steinkühler U, Cruse H (1998) A holistic model for an internal representation to control the movement of a manipulator with redundant degrees of freedom. *Biological Cybernetics* 79: 457-466.

Strauss R, Pichler J (1988) Persistence of orientation towards a temapraily invisible landmark in Drosophila melanogaster. J. Comp. Physiol. A 182: 411-423.

Umiltà MA, Kohler E, Gallese V, Fogassi L, Fadiga L, Keysers C, Rizzolatti G (2001) "I know what you are doing": a neurophysiological study. Neuron 32: 91-101.

Wang X-J (2001) Synaptic reverberation underlying mnemonic persistent activity. *Trends in Neurosciences* 24: 455-463.

Widrow B, Hoff ME (1960) Adaptive switching circuits. In: *1960 WESCON Convention record Part IV*, New York: Institute of Radio Engineers, pp 96-104.

Williams RJ, Zipser D (1989a) A learning algorithm for continually running fully recurrent neural networks. *Neural Computation* 1: 270-280.

Williams RJ, Zipser D (1989b) Experimental analysis of the real-time recurrent learning algorithm. *Connection Science* 1: 87-111.

Wilson H, Cowan J (1973) A mathematical theory of the functional dynamics of cortical and thalamic nervous tissue. *Kybernetik* 13: 55-80.

Wolpert DM, Kawato M (1998) Multiple paired forward and inverse models for motor control. *Neural Networks* 11: 1317-1329.

Xie X, Seung HS (2003) Equivalence of Backpropagation and Contrastive Hebbian Learning in layered networks. *Neural Computation* 15: 441-454.

Zipser D, Kehoe B, Littlewort G, Fuster J (1993) A spiking network model of short-term active memory. *Journal of Neuroscience* 13: 3406-3420.

Captions

Figure 1: a) Schematic drawing of a three-unit recurrent network; a_i is the external input, x_i the recurrent input and w_{ij} are the weights. b) Architecture of one linear IC Unit; s_i(t) is the weighted sum of the recurrent inputs and δ_i(t) the difference between the external input a_i(t) and s_i(t). c) Architecture of one IC Unit with the nonlinear extension (see text for explanation).

Figure 2: Geometrical illustration for the process of training a two-unit network. The axis around which the plane is rotated is denoted by the grey arrow.

Figures

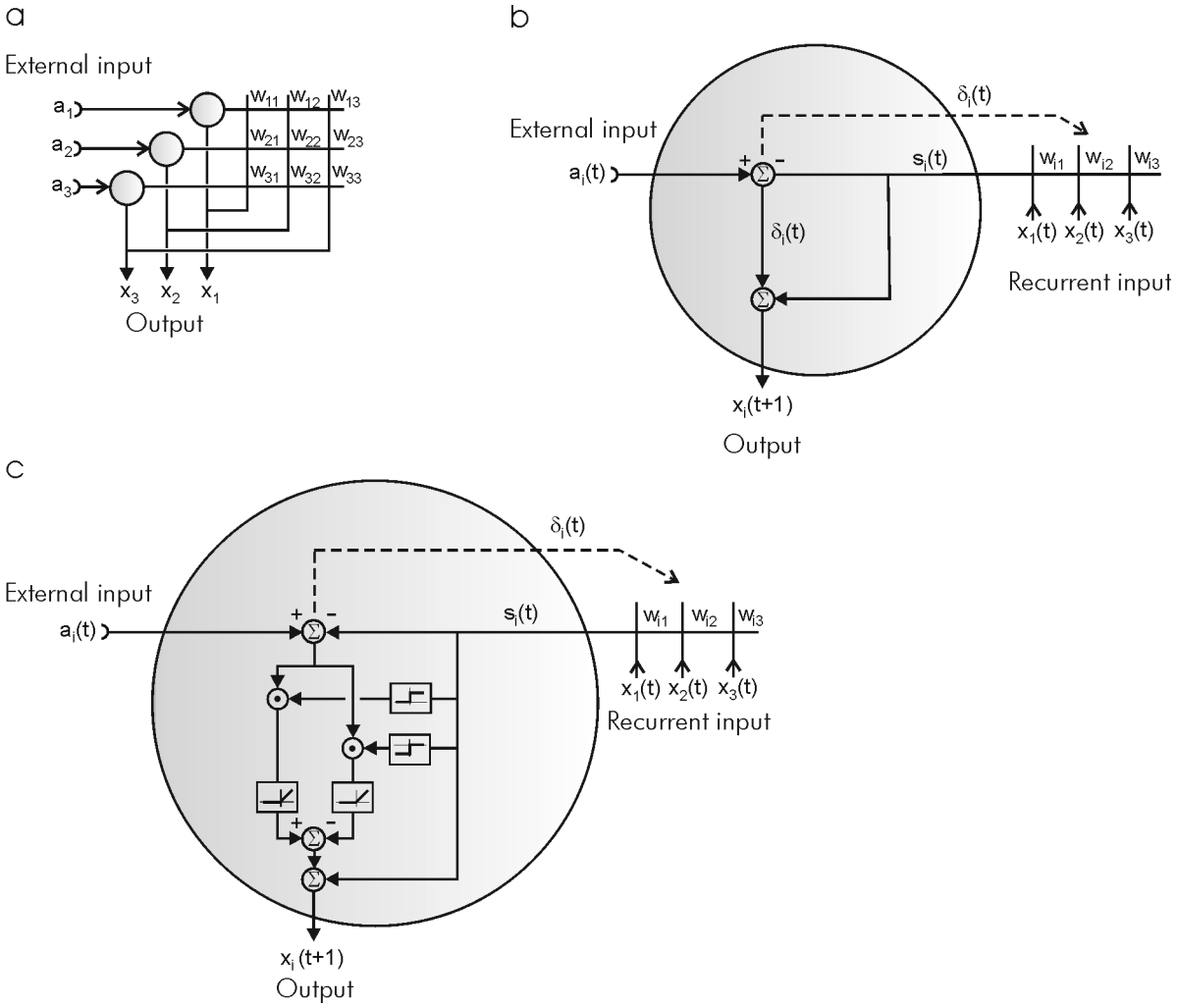

Figure 1

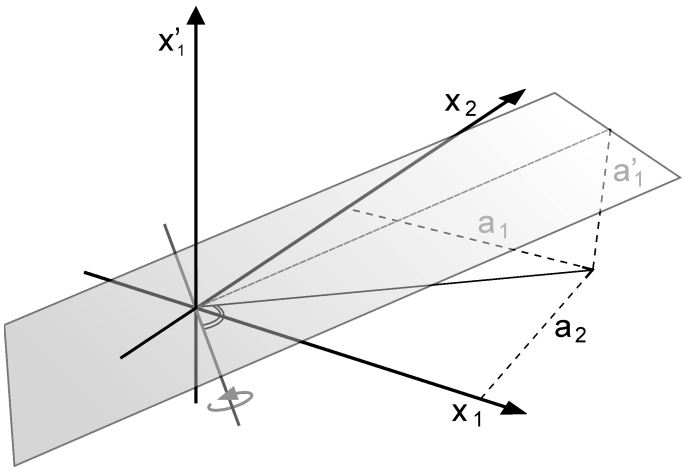

Figure 2