Robust RBF Finite Automata

Michal Šorel [a,1] Jiří Šíma [b,2]

[a] *Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, P.O. Box 18, 182 08 Prague 8, Czech Republic*

[b] *Institute of Computer Science, Academy of Sciences of the Czech Republic, P.O. Box 5, 182 07 Prague 8, Czech Republic*

Abstract

The computational power of recurrent RBF (radial basis functions) networks is investigated. A recurrent network which consists of $O(\sqrt{m \log m})$ RBF units with maximum norm employing any activation function that has different values in at least two nonnegative points, is constructed so as to implement a given deterministic finite automaton with m states. The underlying simulation proves to be robust with regard to bounded analog noise for a large class of smooth activation functions with a special type of inflexion.

Key words: Radial basis function; Neural network; Finite automaton; Boolean circuit; Computational power

1 RBF and Perceptron Neural Networks

The so-called *radial basis functions* (RBF) were first introduced in the context of interpolation problems [25] which are now one of the main research fields in numerical analysis. These functions were later exploited in the design of feedforward neural networks [3,21,24,26] which are called *RBF networks* and represent an important alternative to the classical *perceptron networks* [20]. Unlike the perceptrons that classify unbounded halfspaces, the RBF units are

Email addresses: `michal.sorel@centrum.cz` (Michal Šorel), `sima@cs.cas.cz` (Jiří Šíma).

[1] Partially supported by GA AS CR Grant B2030007.

[2] Partially supported by project LN00A056 of The Ministry of Education of the Czech Republic.

localized around the centers in the areas of bounded width. Hence, also the learning heuristics for the RBF networks may be different and their application fields are usually complementary to those of perceptrons [10].

The computational properties of the traditional perceptron networks seem to be well understood now [33]. The computational power of the finite recurrent discrete-time neural networks with the saturated-linear activation function increases with the Kolmogorov complexity of real weight parameters [2]. For example, such networks with integer weights (corresponding to the binary-state networks with threshold units) are equivalent to finite automata [15], while those with rational weights can simulate arbitrary Turing machines [13,30].

In addition, finer descriptive measures were introduced for the so-called *neuromata* which are binary-state neural networks implementing finite automata [1]. A size-optimal neuromaton with $\Theta(\sqrt{m})$ perceptron units can be constructed so as to simulate a given deterministic finite automaton with m states [12,13]. Similarly, a regular language described by a regular expression of length ℓ can be recognized by a size-optimal neuromaton with $\Theta(\ell)$ perceptrons [32].

These neuromaton constructions were generalized for analog-state recurrent neural nets with a large class of sigmoidal activation functions so that the respective simulations are stable in the sense that potentially infinite computations are robust with respect to the bounded analog noise [4,17,28,31] although such networks are even unable to recognize all regular languages when the analog noise can move states over large distances in the state space [18,29].

One motivation for analog implementations of neuromata issues from real-world applications in which partial prior knowledge (i.e. several automaton transitions) is known and should be refined and completed by training data. However, most of the neural learning heuristics operate in an analog state space (e.g. gradient methods) and hence, the analog neuromaton implementation can serve as an initial network configuration for the underlying learning procedure. The resulting transition rules can also be extracted after the neuromaton has been adapted to training data. This approach has widely been applied [5,6,8,19,22,23,35,37], especially for the second-order neural nets which are also related to the recurrent RBF networks [7]. In addition, the analog implementations of neuromata offer other advantages besides being used in gradient-based training algorithms; they also permit analog VLSI implementation, the foundations necessary for the universal approximation theories, the interpretation of outputs as posteriori probability estimates, etc. [23].

Unlike the perceptron networks, the computational capabilities of recurrent RBF networks have not been fully analyzed yet. Partial theoretical results along this direction have been achieved for hybrid recurrent networks consisting of both the RBF and the perceptron layers which exhibited successful experimental results for inductive inference of regular grammars [7,9].

The present paper investigates the computational power of recurrent RBF networks. In particular, the issue of implementing the deterministic finite automata by recurrent RBF networks is studied. This article is organized as follows. After a brief review of the basic definitions in Section 2, our main construction of a recurrent RBF network of size $O(\sqrt{m \log m})$ units which simulates a given deterministic finite automaton with m states, is described in Section 3. The simulation is achieved here for any activation function that has different values in at least two nonnegative points provided that the maximum norm is used. Thus the recurrent RBF networks prove to be at least as powerful as the finite automata and hence, they can be called *RBF neuromata*.

The idea behind our construction issues from a similar result for perceptron networks [12]. In particular, m automaton states are encoded as Boolean vectors of length $\lceil \log_2 m \rceil$ bits and therefore, the transition function producing the next automaton state can be implemented as a vector Boolean function reading $\lceil \log_2 m \rceil + 1$ bits (i.e. old state and binary input) and writing $\lceil \log_2 m \rceil$ bits (i.e. new state). Furthermore, it is shown that any vector Boolean function of n inputs and p outputs can be implemented by an optimal-size three-layer circuit of $\Theta(\sqrt{p2^n})$ NAND gates (Lemma 1). Finally, each such a multi-input NAND gate (with some of the inputs negated) can be replaced with a RBF unit (Lemma 2). By putting these ingredients together we obtain the desired construction of the RBF neuromaton.

In Section 4, the underlying simulation is shown to be stable for more practical imprecise analog representations of binary states. The result is valid for a large class of smooth activation functions with a special type of inflexion which includes the most common Gaussian activation function. Section 5 illustrates the construction of robust RBF neuromata by a small example, and Section 6 concludes with some open problems. An extended abstract of this paper appeared in [34].

2 Preliminaries

2.1 Recurrent RBF Networks

We will first specify the model of a *recurrent RBF network*. The network consists of s *RBF units*, indexed as $V = \{1, \ldots, s\}$, where s is called the network *size*. The units are connected into an oriented graph representing the *architecture* of the network, in which each edge (i, j) leading from unit i to j is labeled with a real parameter $c_{ji} \in \Re$. Denote by $\vec{c}_j = (c_{ji_1}, \ldots, c_{ji_{n_j}}) \in \Re^{n_j}$

the *center* of RBF unit j where $i_1, \ldots, i_{n_j}$ are all the n_j incoming units incident on j. Each unit $j \in V$ is also associated with its real positive *width* parameter $b_j > 0$.

The *computational dynamics* of a recurrent RBF network determines for each unit $j \in V$ the evolution of its real *state (output)* $y_j^{(t)} \in \Re$ as a function of discrete time t. This establishes the global *network state* $\vec{y}^{(t)} = (y_1^{(t)}, \ldots, y_s^{(t)}) \in \Re^s$ at each time instant $t = 0, 1 \ldots$. At the beginning of a computation, the RBF network is placed in an *initial state* $\vec{y}^{(0)} \in \Re^s$. At discrete time $t \geq 0$, the nonnegative *"excitation"*

$$\xi_j^{(t)} = \frac{\left\| \vec{x}_j^{(t)} - \vec{c}_j \right\|}{b_j} \geq 0 \tag{1}$$

of unit j is proportional to the distance between the *input vector* $\vec{x}_j^{(t)} = (y_{ji_1}^{(t)}, \ldots, y_{ji_{n_j}}^{(t)}) \in \Re^{n_j}$, consisting of the states of units $i_1, \ldots, i_{n_j}$ incident on j, and unit j's center $\vec{c}_j$. For our construction in Section 3 the *maximum norm*

$$\|\vec{x}\| = \|\vec{x}\|_\infty = \max_{i=1,\ldots,n} |x_i| \tag{2}$$

for $\vec{x} = (x_1, \ldots, x_n) \in \Re^n$ is assumed in equation (1).

At the next instant $t+1$, the RBF units $j \in \alpha_{t+1}$ from a selected subset $\alpha_{t+1} \subseteq V$ compute their new outputs $y_j^{(t+1)}$ in parallel by applying an *activation function* $\sigma : \Re_0^+ \longrightarrow \Re$ (domain $\Re_0^+$ represents the set of nonnegative real numbers) to excitations $\xi_j^{(t)} \geq 0$, i.e.

$$y_j^{(t+1)} = \sigma\left(\xi_j^{(t)}\right), \quad j \in \alpha_{t+1}, \tag{3}$$

while the remaining RBF units $j \in V \setminus \alpha_{t+1}$ do not change their states, that is $y_j^{(t+1)} = y_j^{(t)}$ for $j \notin \alpha_{t+1}$. Thus, α_{t+1} denotes the set of RBF units whose outputs are updated at time instant $t+1$. For instance, α_t in feedforward architectures coincides with the tth layer which evaluates its outputs in parallel (cf. Section 2.3). A typical example of the activation function employed by the RBF networks is the *Gaussian function*

$$\sigma_G(\xi) = e^{-\xi^2} \tag{4}$$

for $\xi \geq 0$. In this way the new state $\vec{y}^{(t+1)}$ of the RBF network at time $t+1$ is determined.

2.2 RBF Neuromata

The computational power of recurrent networks has been studied analogously to the traditional models of computations so that the networks are exploited as acceptors of languages $L \subseteq \{0,1\}^\star$ over the binary alphabet. For the finite RBF networks that are to recognize regular languages, i.e. the so-called *RBF neuromata*, the following input/output protocol can be defined. Since the activation function σ can produce arbitrary real outputs, an *analog representation* $\varrho : \{0,1\} \longrightarrow \Re$ such that $\varrho(0) \neq \varrho(1)$, is introduced to encode the binary values 0 and 1 by two different analog states $\varrho(0)$ and $\varrho(1)$ of RBF units, respectively. The analog representation will also be used for Boolean vectors, i.e. $\vec{\varrho}(z_1, \ldots, z_n) = (\varrho(z_1), \ldots, \varrho(z_n))$ for $\vec{z} = (z_1, \ldots, z_n) \in \{0,1\}^n$, $n \geq 1$.

Thus a binary input string $\vec{w} = w_1 \ldots w_k \in \{0,1\}^k$ of an arbitrary length $k \geq 0$ is sequentially presented to the RBF neuromaton bit by bit via an *input unit* $\mathrm{inp} \in V$. The state of this unit is externally set (and e.g. clamped) to the respective input-bit representations at prescribed time instants, regardless of any influence from the remaining RBF units in the network, i.e.

$$y_{\mathrm{inp}}^{(\tau(i-1))} = \varrho(w_i) \tag{5}$$

for $i = 1, \ldots, k$ where an integer constant $\tau \geq 1$ is the *time overhead* for processing a single input bit. Then an *output unit* $\mathrm{out} \in V$ signals at time τk whether the input word $\vec{w}$ belongs to underlying language L, i.e.

$$y_{\mathrm{out}}^{(\tau k)} = \begin{cases} \varrho(1) & \text{for } \vec{w} \in L \\ \varrho(0) & \text{for } \vec{w} \notin L\,. \end{cases} \tag{6}$$

Similar definitions of language recognition by perceptron neuromata appeared in [1,13,30,32].

2.3 Boolean Circuits

For the construction of RBF neuromata in Section 3, the implementation of an arbitrary vector Boolean function by an optimal-size circuit will be exploited. The model of *Boolean circuits* [36] (hereafter briefly *circuits*) coincides with the model of *feedforward networks* whose architectures are acyclic graphs, and the units are more commonly called *gates*. By induction on t, gates in a circuit can be grouped in a unique minimal way into a sequence of $d+1$ pairwise disjoint *layers* $\alpha_0, \ldots, \alpha_t, \ldots, \alpha_d \subseteq V$ as follows. The zero, or *input layer* α_0 consists of n external inputs. For the induction step, assume that layers $\alpha_0, \ldots, \alpha_t$ $(0 \leq t < d)$ have already been identified. Next layer α_{t+1}

then includes *all* the gates whose *all* inputs come from the preceding layers $\alpha_0, \ldots, \alpha_t$. Finally, the last, or *output layer* α_d is composed of p output gates. The number of layers d excluding the input one is called the circuit *depth*.

Computation in a circuit then proceeds from the input layer via the intermediate layers up to the output layer. Initially, the outputs from the input layer are set to represent an external input. In a general step, supposing all the outputs of gates have been determined up to a certain layer $t < d$, the gates in the next layer α_{t+1} then compute their outputs in parallel. For this purpose, to each non-input gate j a logical function is assigned whose n_j Boolean variables $z_1, \ldots, z_{n_j} \in \{0,1\}$ correspond to the outputs from the incoming gates incident on j. The gates that compute logical NAND functions over multiple *literals* will be employed in Section 3. In particular, inputs $z_1, \ldots, z_{n_j}$ to gate j are either negated or not, which produces literals $\lambda_{a_{j1}}(z_1), \ldots, \lambda_{a_{jn_j}}(z_{n_j})$ in variables $z_1, \ldots, z_{n_j}$ according to parameters $a_{j1} \ldots, a_{jn_j} \in \{0,1\}$ local to j as follows:

$$\lambda_a(z) = \begin{cases} z & \text{for } a = 1 \\ \bar{z} & \text{for } a = 0 \end{cases} \tag{7}$$

where $a = 0$ defines negated variable $\bar{z}$ whereas $a = 1$ represents variable z itself. Then an NAND gate outputs

$$\mathrm{NAND}\left(\lambda_{a_{j1}}(z_1), \ldots, \lambda_{a_{jn_j}}(z_{n_j})\right) = 0 \quad \text{iff} \quad z_i = a_{ji} \ \text{ for all } \ i = 1, \ldots, n_j \,.$$

Thus the output of any gate is computed by evaluating the associated logical function. Eventually, the outputs from output layer α_d represent the result of the computation. In this way a vector Boolean function $\vec{f} : \{0,1\}^n \longrightarrow \{0,1\}^p$ is evaluated in a number of parallel steps equal to depth d of the underlying circuit.

3 The construction of RBF neuromaton

In this section our main construction of the RBF neuromaton simulating a given deterministic finite automaton will be described. For this purpose, the following two lemmas are first proved. In Lemma 1, a circuit composed of NAND gates is constructed so as to implement a given vector Boolean function, which is very similar to the result presented in [11, Lemma 5]. In addition, a counting argument very similar to the one described in [11, Lemma 6] provides the proof that in the worst case, the size of the circuit from Lemma 1 cannot be reduced even for unbounded depth, and thus the construction in Lemma 1 proves to be optimal with regard to the circuit size. Subsequent Lemma 2 then presents the implementation of any NAND gate over multiple literals by the RBF unit within an analog representation of Boolean values.

Lemma 1 *Any vector Boolean function $\vec{f}:\{0,1\}^n \longrightarrow \{0,1\}^p$ can be implemented by a (optimal-size) Boolean circuit of depth 3 with $O(\sqrt{p2^n})$ gates computing logical NAND functions over multiple literals.*

PROOF. Shannon's decomposition [27] (see [36]) can be used to express each component $f_j:\{0,1\}^n \longrightarrow \{0,1\}$ of vector Boolean function $\vec{f}=(f_1,\ldots,f_p)$ as a disjunction of 2^r terms for any choice of integer parameter $1 \le r \le n$:

$$\begin{aligned} & f_j\left(z_1,\ldots,z_n\right) \\ & = \bigvee_{(a_1,\ldots,a_r)\in\{0,1\}^r} \left(\bigwedge_{i=1}^{r} \lambda_{a_i}(z_i) \;\wedge \bigwedge_{(a_{r+1},\ldots,a_n)\in N_{j,(a_1,\ldots,a_r)}} \left(\overline{\bigwedge_{i=r+1}^{n} \lambda_{a_i}(z_i)} \right) \right) \end{aligned} \quad (8)$$

for every $z_1,\ldots,z_n \in \{0,1\}^n$ where notation (7) is used and

$$N_{j,(a_1,\ldots,a_r)} = \left\{ (a_{r+1},\ldots,a_n) \in \{0,1\}^{n-r}\,;\; f_j(a_1,\ldots,a_n)=0 \right\}\,. \quad (9)$$

Clearly, within each term of disjunction (8) that is determined by parameter vector $(a_1,\ldots,a_r)\in\{0,1\}^r$, the first conjunction checks whether the first r variables $z_1,\ldots,z_r$ have values $a_1,\ldots,a_r$ while the second conjunction verifies whether the values of the remaining $n-r$ variables $z_{r+1},\ldots,z_n$ do not belong to set $N_{j,(a_1,\ldots,a_r)}$ corresponding to zero values of f_j as defined in (9). Formula (8) can further be rewritten by using De Morgan's laws:

$$\begin{aligned} & f_j\left(z_1,\ldots,z_n\right) \\ & = \overline{\bigwedge_{(a_1,\ldots,a_r)\in\{0,1\}^r} \left(\overline{\bigwedge_{i=1}^{r} \lambda_{a_i}(z_i) \;\wedge \bigwedge_{(a_{r+1},\ldots,a_n)\in N_{j,(a_1,\ldots,a_r)}} \left(\overline{\bigwedge_{i=r+1}^{n} \lambda_{a_i}(z_i)} \right)} \right)} \end{aligned} \quad (10)$$

which, in fact, contains only the logical NAND functions of literals. Therefore, formula (10) can directly be used to construct the desired Boolean circuit for $\vec{f}:\{0,1\}^n \longrightarrow \{0,1\}^p$ as follows.

The third, output layer α_3 of the underlying depth-3 circuit is composed of the p NAND gates, one for each component f_j of $\vec{f}=(f_1,\ldots,f_p)$ corresponding to the conjunction under the outer negation in formula (10). The inputs to these output gates are taken from the second layer α_2 that consists of $p2^r$ NAND gates. That is, for each component f_j ($j=1,\ldots,p$) there are 2^r gates (unique to f_j) in α_2, corresponding to all possible parameter vectors $(a_1,\ldots,a_r)\in\{0,1\}^r$. Furthermore, the NAND gates from α_2 implement the conjunctions under the intermediate negation in formula (10) corresponding to the negation of the original terms of disjunction (8). Hence, the inputs to the NAND gates in α_2 include the literals in variables $z_1,\ldots,z_r$ corresponding to the external inputs in input layer α_0. The remaining inputs to α_2 are taken from the first layer α_1 which is composed of (at most) 2^{n-r} NAND gates, one for each possible parameter vector $(a_{r+1}, \ldots, a_n) \in \{0,1\}^{n-r}$ that, depending on sets $N_{j,(a_1,\ldots,a_r)}$ for $j = 1, \ldots, p$, may appear in the conjunctions under the innermost negation in formula (10). Thus, the inputs to the NAND gates in α_1 are literals in variables $z_{r+1}, \ldots, z_n$ corresponding to the external inputs in input layer α_0. This completes the circuit construction.

In addition, the size $s(r)$ of the underlying circuit will be determined as a function of parameter r by summing up the numbers of gates in particular layers:

$$s(r) = p + p2^r + 2^{n-r} \tag{11}$$

which reaches its minimum (cf. [11, Lemma 5]) for

$$r = \frac{n - \log_2 p}{2} \,. \tag{12}$$

By substituting (12) in (11) the desired circuit size $O(\sqrt{p2^n})$ is achieved. $\square$

Lemma 2 *Assume a real function σ is defined in at least two different non-negative points $0 \leq \xi_1 < \xi_2$ so that $\sigma(\xi_1) \neq \sigma(\xi_2)$. Denote by $g : \{0,1\}^n \longrightarrow \{0,1\}$ a Boolean function defined as the logical NAND function over $n \geq 1$ literals, i.e.*

$$g(z_1, \ldots, z_n) = \mathrm{NAND}\left(\lambda_{a_1}(z_1), \ldots, \lambda_{a_n}(z_n)\right) \tag{13}$$

for a parameter vector $\vec{a} = (a_1, \ldots, a_n) \in \{0,1\}^n$ according to notation (7). Then the n-input RBF unit with the center $\vec{c} = (c_1, \ldots, c_n) \in \Re^n$ such that

$$c_i = \begin{cases} \frac{\xi_2\sigma(\xi_2) - \xi_1\sigma(\xi_1)}{\xi_2 - \xi_1} = c & \text{for } a_i = 1 \\ \frac{\xi_2\sigma(\xi_1) - \xi_1\sigma(\xi_2)}{\xi_2 - \xi_1} = \bar{c} & \text{for } a_i = 0 \end{cases} \qquad i = 1, \ldots, n \,, \tag{14}$$

and width

$$b = \frac{|\sigma(\xi_2) - \sigma(\xi_1)|}{\xi_2 - \xi_1} > 0 \,, \tag{15}$$

employing the maximum norm and activation function σ, computes g within the analog representation

$$\varrho(z) = \begin{cases} \sigma(\xi_1) & \text{for } z = 0 \\ \sigma(\xi_2) & \text{for } z = 1 \,, \end{cases} \tag{16}$$

i.e.

$$\sigma\left(\frac{\|\vec{\varrho}(\vec{z}) - \vec{c}\|}{b}\right) = \varrho(g(\vec{z})) \tag{17}$$

for every $\vec{z} \in \{0,1\}^n$.

PROOF. It follows from (14) and (15) that

$$|\sigma(\xi_1)-\bar{c}| = |\sigma(\xi_2)-c| = b\xi_1 < b\xi_2 = |\sigma(\xi_1)-c| = |\sigma(\xi_2)-\bar{c}|\,. \tag{18}$$

Furthermore, according to the definition of NAND function, $\vec{a} \in \{0,1\}^n$ represents the only input that yields $g(\vec{a}) = 0$. Thus consider first the case when $\vec{z} = \vec{a}$ for which

$$\sigma\left(\frac{\|\vec{\varrho}(\vec{a})-\vec{c}\|}{b}\right) = \sigma\left(\frac{b\xi_1}{b}\right) = \sigma(\xi_1) = \varrho(0) \tag{19}$$

holds according to (18) and (16) since

$$|\varrho(a_i)-c_i| = \begin{cases} |\sigma(\xi_1)-\bar{c}| & \text{for } a_i = 0 \\ |\sigma(\xi_2)-c| & \text{for } a_i = 1 \end{cases} \qquad i = 1,\ldots,n \tag{20}$$

follows from (14) and (16). On the other hand, for any $\vec{z} \in \{0,1\}^n$ different from $\vec{a}$ there exists $1 \leq j \leq n$ such that $z_j \neq a_j$, that is either $z_j = 0$ and $a_j = 1$ implying $c_j = c$ according to (14), or $z_j = 1$ and $c_j = \bar{c}$, which gives

$$|\varrho(z_j)-c_j| = \begin{cases} |\sigma(\xi_1)-c| & \text{for } z_j = 0 \\ |\sigma(\xi_2)-\bar{c}| & \text{for } z_j = 1 \end{cases} \tag{21}$$

by using (16). Hence,

$$\sigma\left(\frac{\|\vec{\varrho}(\vec{z})-\vec{c}\|}{b}\right) = \sigma\left(\frac{b\xi_2}{b}\right) = \sigma(\xi_2) = \varrho(1) \tag{22}$$

for any $\vec{z} \neq \vec{a}$ due to (18) and (16). $\Box$

Theorem 3 *Let σ be a real function that assigns different values $\sigma(\xi_1) \neq \sigma(\xi_2)$ to at least two nonnegative points $0 \leq \xi_1 < \xi_2$ which establish the analog representation ϱ by formula (16). Given a deterministic finite automaton with m states, an equivalent RBF neuromaton of size $O(\sqrt{m \log m})$ units with maximum norm and activation function σ can be constructed so that it recognizes the same regular language for the analog representation ϱ within the time overhead $\tau = 3$ per each input bit.*

PROOF. Set Q of m states of a given deterministic finite automaton can be arbitrarily enumerated so that each $q \in Q$ is binary encoded within $\lceil \log_2 m \rceil$ bits. The underlying transition function $\delta : Q \times \{0,1\} \longrightarrow Q$ producing the new automaton state $q_{\text{new}} = \delta(q_{\text{old}}, z) \in Q$ from the old state $q_{\text{old}} \in Q$ and input bit $z \in \{0,1\}$ can be viewed as a vector Boolean function $\vec{f_\delta} : \{0,1\}^{\lceil \log_2 m \rceil + 1} \longrightarrow \{0,1\}^{\lceil \log_2 m \rceil}$ in terms of binary encoding of states. By

Lemma 1 vector function $\vec{f_\delta}$ can be implemented by an optimal-size Boolean circuit of depth 3 with $\Theta(\sqrt{m \log m})$ gates computing the NAND functions over multiple literals. In addition, according to Lemma 2 the NAND gates in this circuit can be replaced by the RBF units with the maximum norm and activation function σ within analog representation ϱ. Hence, the automaton transition function $\vec{f_\delta}$ can be implemented by a three-layered feedforward RBF network of size $O(\sqrt{m \log m})$ units, which is further used for constructing the respective RBF neuromaton as follows.

Thus, recurrent connections are introduced into the underlying three-layered feedforward RBF network so that the inputs to the first layer are taken from the third-layer units storing the current neuromaton state within the analog representation ϱ. In addition, a single external input unit "inp" is introduced whose state is updated by presenting analog representation of bit by bit of input string $\vec{w} = w_1 \ldots w_k$ once every $\tau = 3$ time instants as described in (5). At the beginning of an RBF neuromaton computation, the third-layer units encode the initial automaton state $q_0 \in Q$ and $y_{\mathrm{inp}}^{(0)} = \varrho(w_1)$ represents the first input bit. Set α_t of the RBF units that update their states according to (3) at time instant $1 \leq t \leq \tau k$ coincides with the uth layer for $t = \tau(i-1)+u$ when the ith bit w_i of input string $\vec{w}$ is being processed. This means that the binary encoding of the new state is computed within the time overhead of three parallel updates in which each of the three layers takes its part. Eventually, at time instant τk, the input string $\vec{w}$ of length k is read and accepted iff the third-layer units encode an accepting automaton state. This is indicated by an additional bit in the binary encoding of automaton states corresponding to the output unit "out" whose state $y_{\mathrm{out}}^{(\tau k)} = \varrho(1)$ iff $\vec{w}$ is accepted according to (6). This completes the construction of the desired RBF neuromaton. $\Box$

4 Robust RBF Neuromata

In Section 3 the RBF neuromaton is implemented only for precise analog representation of binary values assuming noise-free operations over real numbers. However, the infinite precision of real number representation is far from being realistic and an analog noise (e.g. rounding errors) could deteriorate long RBF neuromaton computations. Therefore, in this section the underlying construction is analyzed to find out conditions under which the respective RBF neuromaton is robust provided that the analog representation ϱ of binary values 0 and 1 is released to be within certain disjoint small intervals around real states $\varrho(0)$ and $\varrho(1)$, respectively.

For the construction of a RBF neuromaton that is robust against bounded analog noise it suffices to show that the implementation of the NAND gates by the RBF units according to Lemma 2 preserves the prescribed precision of analog representation ϱ. More precisely, the underlying computation of Boolean function g by the RBF unit (with activation function σ defined now in a sufficiently large domain) is said to be *ε-stable*,

$$0 < \varepsilon < \frac{|\varrho(1) - \varrho(0)|}{2}\,, \tag{23}$$

if for every binary input $\vec{z} \in \{0,1\}^n$ and any $\vec{x} \in \Re^n$ of its analog representation within precision ε such that $|\varrho(z_i) - x_i| \leq \varepsilon$ for $i = 1, \ldots, n$, the output precision of the RBF unit is preserved, i.e.

$$\|\vec{\varrho}(\vec{z}) - \vec{x}\| \leq \varepsilon \quad \longrightarrow \quad \left| \sigma\left(\frac{\|\vec{\varrho}(\vec{z}) - \vec{c}\|}{b}\right) - \sigma\left(\frac{\|\vec{x} - \vec{c}\|}{b}\right) \right| \leq \varepsilon\,. \tag{24}$$

A large class of differentiable activation functions for which Lemma 2 provides a robust implementation of the NAND gates by the RBF units will now be characterized.

Theorem 4 *Let σ be a real function that is differentiable within sufficiently large neighborhood*

$$D(\xi_1, \xi_2, \varepsilon) = \left[\xi_1 - \frac{\varepsilon}{b}, \xi_1 + \frac{\varepsilon}{b}\right] \cup \left[\xi_2 - \frac{\varepsilon}{b}, \xi_2 + \frac{\varepsilon}{b}\right] \tag{25}$$

of two positive points $0 < \xi_1 < \xi_2$ where $\sigma(\xi_1) \neq \sigma(\xi_2)$, parameter b is introduced in (15), and ε is a real number satisfying

$$0 < \varepsilon < \min\left(\xi_1 b\,,\ \frac{|\sigma(\xi_2) - \sigma(\xi_1)|}{2}\right)\,. \tag{26}$$

Then the computation of Boolean function g by the RBF unit as described in Lemma 2 is ε-stable if

$$|\sigma'(\xi)| \leq b = \frac{|\sigma(\xi_2) - \sigma(\xi_1)|}{\xi_2 - \xi_1} \qquad \textit{for all } \xi \in D(\xi_1, \xi_2, \varepsilon) \tag{27}$$

and is not ε-stable if $|\sigma'(\xi)| > b$ for all $\xi \in D(\xi_1, \xi_2, \varepsilon)$.

PROOF. First suppose that σ satisfies (27) and let $\vec{z} \in \{0,1\}^n$ and $\vec{x} \in \Re^n$ be such that

$$\|\vec{\varrho}(\vec{z}) - \vec{x}\| \leq \varepsilon\,. \tag{28}$$

It follows from the triangular inequality that

$$\|\vec{\varrho}(\vec{z}) - \vec{c}\| - \|\vec{\varrho}(\vec{z}) - \vec{x}\| \leq \|\vec{x} - \vec{c}\| \leq \|\vec{\varrho}(\vec{z}) - \vec{c}\| + \|\vec{x} - \vec{\varrho}(\vec{z})\| \tag{29}$$

which, divided by $b > 0$, gives

$$\frac{\|\vec{\varrho}(\vec{z}) - \vec{c}\|}{b} - \frac{\varepsilon}{b} \leq \frac{\|\vec{x} - \vec{c}\|}{b} \leq \frac{\|\vec{\varrho}(\vec{z}) - \vec{c}\|}{b} + \frac{\varepsilon}{b} \tag{30}$$

according to (28). Recall from the proof of Lemma 2 that

$$\frac{\|\vec{\varrho}(\vec{z})-\vec{c}\|}{b}=\begin{cases}\xi_1 & \text{for } \vec{z}=\vec{a}\\ \xi_2 & \text{for } \vec{z}\neq\vec{a}\end{cases} \tag{31}$$

which implies

$$\frac{\|\vec{x}-\vec{c}\|}{b}\in D(\xi_1,\xi_2,\varepsilon) \tag{32}$$

according to (30). Therefore by the mean value theorem there exists $\zeta \in D(\xi_1,\xi_2,\varepsilon)$ such that

$$\sigma\left(\frac{\|\vec{\varrho}(\vec{z})-\vec{c}\|}{b}\right)-\sigma\left(\frac{\|\vec{x}-\vec{c}\|}{b}\right)=\sigma'(\zeta)\left(\frac{\|\vec{\varrho}(\vec{z})-\vec{c}\|}{b}-\frac{\|\vec{x}-\vec{c}\|}{b}\right) \tag{33}$$

which gives the desired ε-stability (24) since $|\sigma'(\zeta)|\leq b$ from (27) and

$$\left|\,\|\vec{\varrho}(\vec{z})-\vec{c}\|-\|\vec{x}-\vec{c}\|\,\right|\leq\varepsilon \tag{34}$$

according to (29) and (28).

On the other hand, suppose $|\sigma'(\xi)|>b$ for all $\xi\in D(\xi_1,\xi_2,\varepsilon)$ and let $\vec{z}\in\{0,1\}^n$. Choose

$$\vec{x}_0=\vec{\varrho}(\vec{z})+\varepsilon\frac{\vec{\varrho}(\vec{z})-\vec{c}}{\|\vec{\varrho}(\vec{z})-\vec{c}\|} \tag{35}$$

for which $\|\vec{\varrho}(\vec{z})-\vec{x}_0\|=\varepsilon$, that is $\vec{x}_0\in\Re^n$ is still an analog representation of $\vec{z}$ within precision ε. In addition,

$$\|\vec{x}_0-\vec{c}\|=\left\|\vec{\varrho}(\vec{z})-\vec{c}+\varepsilon\frac{\vec{\varrho}(\vec{z})-\vec{c}}{\|\vec{\varrho}(\vec{z})-\vec{c}\|}\right\|=\left(1+\frac{\varepsilon}{\|\vec{\varrho}(\vec{z})-\vec{c}\|}\right)\|\vec{\varrho}(\vec{z})-\vec{c}\| \tag{36}$$

by (35), which gives

$$\frac{\|\vec{\varrho}(\vec{z})-\vec{c}\|}{b}-\frac{\|\vec{x}_0-\vec{c}\|}{b}=\frac{\varepsilon}{b}\,. \tag{37}$$

Hence,

$$\frac{\|\vec{x}_0-\vec{c}\|}{b}\in D(\xi_1,\xi_2,\varepsilon) \tag{38}$$

according to (31). Thus again by the mean value theorem (33) there exists $\zeta_0\in D(\xi_1,\xi_2,\varepsilon)$ such that

$$\left|\sigma\left(\frac{\|\vec{\varrho}(\vec{z})-\vec{c}\|}{b}\right)-\sigma\left(\frac{\|\vec{x}_0-\vec{c}\|}{b}\right)\right|=|\sigma'(\zeta_0)|\,\frac{\varepsilon}{b}>\varepsilon \tag{39}$$

due to (37) and $|\sigma'(\zeta_0)|>b$, which means that the computation of g is not ε-stable according to (24). $\square$

Condition (27) that characterizes the differentiable activation functions for which the constructed RBF neuromaton is robust against bounded analog noise includes parameter b depending on the function values again. In the following lemma a useful characterization of smooth functions, which implies (27), is formulated. In particular, such functions have a special type of an inflexion point at which they pass from convex to concave part when they are increasing while the decreasing ones change conversely.

Lemma 5 *Let σ be a real function that has continuous derivative within $(\alpha, \beta) \subseteq [0, \infty)$. Assume that there exist $\xi_0 \in (\alpha, \beta)$ with $\sigma'(\xi_0) \neq 0$ and $\Delta > 0$ such that σ is either convex within $(\xi_0 - \Delta, \xi_0)$ and concave within $(\xi_0, \xi_0 + \Delta)$ if $\sigma'(\xi_0) > 0$, or concave within $(\xi_0 - \Delta, \xi_0)$ and convex within $(\xi_0, \xi_0 + \Delta)$ when $\sigma'(\xi_0) < 0$. Then there exist $\alpha < \xi_1 < \xi_2 < \beta$ such that $\sigma(\xi_1) \neq \sigma(\xi_2)$ and $\varepsilon > 0$ satisfying (26), for which condition (27) in Theorem 4 holds, i.e. for all $\xi \in D(\xi_1, \xi_2, \varepsilon)$*

$$|\sigma'(\xi)| \leq \frac{|\sigma(\xi_2) - \sigma(\xi_1)|}{\xi_2 - \xi_1} \,. \tag{40}$$

PROOF. Assume that $\sigma'(\xi_0) > 0$ while for $\sigma'(\xi_0) < 0$ the argument is analogous. Since σ' is continuous there exists $0 < \Delta' < \Delta$ such that $\sigma'(\xi) > 0$ for all $\xi \in (\xi_0 - \Delta', \xi_0 + \Delta') \subseteq (\alpha, \beta)$. According to the assumption, σ' is nondecreasing within $(\xi_0 - \Delta', \xi_0)$ and nonincreasing within $(\xi_0, \xi_0 + \Delta')$, and hence σ' has its maximum at ξ_0.

Consider first the case when $\xi_0 \in [\eta, \vartheta] \subseteq (\xi_0 - \Delta', \xi_0 + \Delta')$ is contained within a nontrivial interval (i.e. $\eta < \vartheta$) such that $\sigma'(\xi) = \sigma'(\xi_0)$ for all $\xi \in [\eta, \vartheta]$. Then define $\xi_1, \xi_2 \in [\eta, \vartheta]$, e.g.

$$\xi_1 = \eta + \frac{\varepsilon}{\sigma'(\xi_0)} \,, \qquad \xi_2 = \vartheta - \frac{\varepsilon}{\sigma'(\xi_0)} \tag{41}$$

where

$$\varepsilon = \frac{\sigma'(\xi_0)(\vartheta - \eta)}{5} \,, \tag{42}$$

for which

$$\sigma'(\xi) = \sigma'(\xi_0) = \frac{\sigma(\xi_2) - \sigma(\xi_1)}{\xi_2 - \xi_1} = b \tag{43}$$

for all $\xi \in [\eta, \vartheta]$ and $\sigma(\xi_1) \neq \sigma(\xi_2)$, since σ' is positive and constant here. It follows from (41)–(43) that $D(\xi_1, \xi_2, \varepsilon) \subseteq [\eta, \vartheta]$ which implies (40) according to (43).

In the opposite case when $\sigma'(\xi) \neq \sigma'(\xi_0)$ for all $\xi \in (\xi_0 - \Delta', \xi_0) \cup (\xi_0, \xi_0 + \Delta')$, there exist $\xi_1 \in (\xi_0 - \Delta', \xi_0)$ and $\xi_2 \in (\xi_0, \xi_0 + \Delta')$ such that $\sigma'(\xi_1) = \sigma'(\xi_2) = v$, and σ' is strictly increasing within (ξ_1, ξ_0) and strictly decreasing within

(ξ_0, ξ_2), since σ' is continuous. This means $\sigma'(\xi) > v$ for all $\xi \in (\xi_1, \xi_2)$ and again $\sigma(\xi_1) \neq \sigma(\xi_2)$ because σ is increasing within $(\xi_0 - \Delta', \xi_0 + \Delta')$. Hence,

$$v(\xi_2 - \xi_1) < \int_{\xi_1}^{\xi_2} \sigma'(\xi) d\xi = \sigma(\xi_2) - \sigma(\xi_1)\,, \tag{44}$$

which implies

$$0 < \sigma'(\xi_1) = \sigma'(\xi_2) = v < \frac{\sigma(\xi_2) - \sigma(\xi_1)}{\xi_2 - \xi_1} = b\,. \tag{45}$$

Since σ' is continuous there exists $\varepsilon > 0$ satisfying (26) such that $|\sigma'(\xi)| < b$ also for all $\xi \in D(\xi_1, \xi_2, \varepsilon)$. $\square$

5 An Example of the RBF Neuromaton Construction

In this section, the presented construction of the RBF neuromaton from a given deterministic finite automaton and its simulation which is robust against bounded analog noise will be illustrated by a small example. Consider a regular language described by regular expression $(10)^*1$ containing words $1, 101, 10101, 1010101$, etc. The transition function δ of a deterministic finite automaton recognizing this regular language is given by the state diagram depicted in Figure 1. This finite automaton includes initial state q_0, rejecting "halting" state q_1, and the only accepting state q_2, which are binary encoded by e.g. 00, 01, and 10, respectively. Note that 1 in the first bit within this code indicates the accepting state. The corresponding vector Boolean transition function $\vec{f_\delta} : \{0,1\}^3 \longrightarrow \{0,1\}^2$ with $n = 3$ inputs and $p = 2$ outputs for the underlying binary encoding of states, which was introduced in Theorem 3,

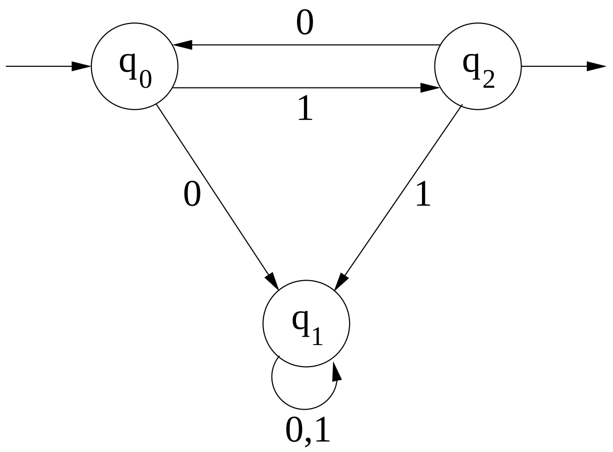

Fig. 1. A deterministic finite automaton for $(10)^*1$.

Table 1
A corresponding vector Boolean transition function $\vec{f_\delta}$.

old state code		input	new state code	
z_1	z_2	z_3	$f_1(z_1,z_2,z_3)$	$f_2(z_1,z_2,z_3)$
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	0	0
1	0	1	0	1

is defined in Table 1. According to Lemma 1, function $\vec{f_\delta}$ is then implemented by a circuit with 10 NAND gates (optimal choice of $r = (3 - \log_2 2)/2 = 1$ is computed by (12)) which is depicted in Figure 2 where as usual the semicircles and small circles denote the conjunctions and negations, respectively.

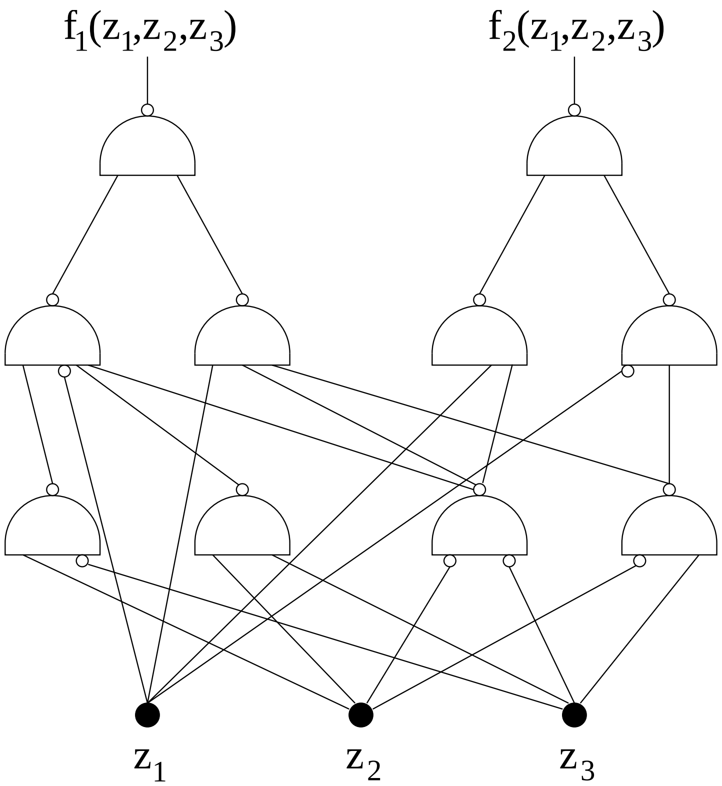

Fig. 2. An NAND circuit for $\vec{f_\delta}$.

By Theorem 3, this circuit is further transformed into an RBF neuromaton in Figure 3 where the original layers of the circuit corresponding to sets of

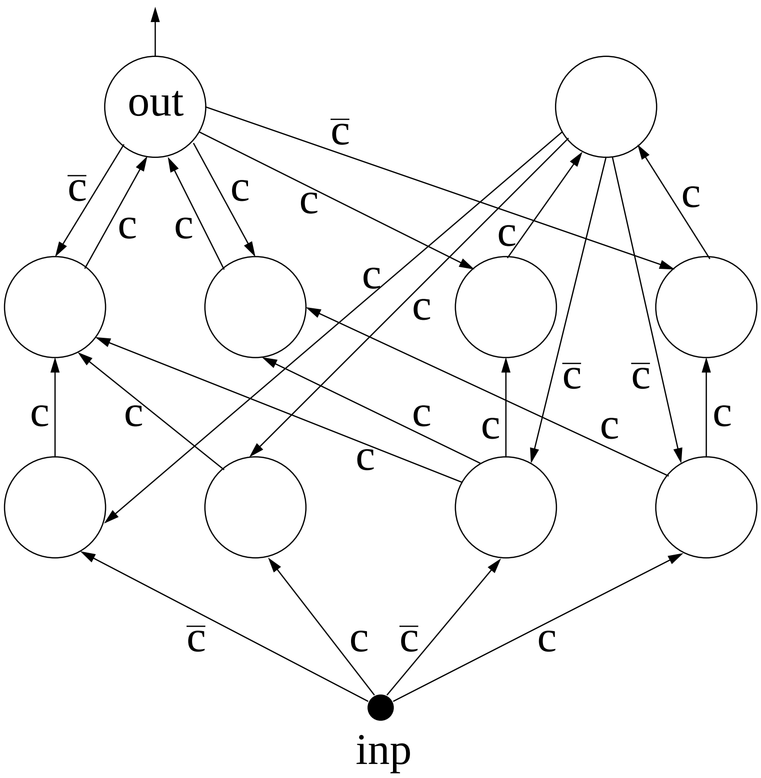

Fig. 3. An RBF neuromaton for $(10)^*1$.

updated units α_t are still noticeable (cf. Figure 2) although the neuromaton architecture is actually recurrent. Thus the states of the units in the "top" layer store the current automaton state within the analog representation of its encoding. The edges within the architecture are labeled with parameters $c, \bar{c}$ determining the centers of particular RBF units, whose values are specified below in (48)–(50) together with the width parameter b that is the same here for all the RBF units in the network.

Observe that if activation function σ has the first three derivatives in its inflexion point $\xi_0 > 0$ satisfying $\sigma''(\xi_0) = 0$ and $\sigma'(\xi_0)\sigma'''(\xi_0) < 0$, then the type of this inflexion corresponds to that described by the assumption of Lemma 5. This verifies that the most common Gaussian activation function (4) can be employed for the robust implementation of RBF neuromata according to Theorem 3. In particular, $\xi_0 = 1/\sqrt{2}$ is the desired inflexion point, i.e. $\sigma''_G(\xi_0) = 0$ and $\sigma'_G(\xi_0)\sigma'''_G(\xi_0) < 0$. It follows from Theorem 4 that e.g.

$$\xi_1 = \sqrt{\ln \frac{10}{9}} \doteq 0.325\,, \qquad \xi_2 = \sqrt{\ln 5} \doteq 1.269\,, \tag{46}$$

determining the following parameters of the RBF neuromaton:

$$\varrho(0) = \sigma_G(\xi_1) = 0.9\,, \qquad \varrho(1) = \sigma_G(\xi_2) = 0.2\,, \tag{47}$$

$$c = \frac{2\sqrt{\ln 5} - 9\sqrt{\ln \frac{10}{9}}}{10\left(\sqrt{\ln 5} - \sqrt{\ln \frac{10}{9}}\right)} \doteq -0.041\,, \tag{48}$$

$$\bar{c} = \frac{9\sqrt{\ln 5} - 2\sqrt{\ln \frac{10}{9}}}{10\left(\sqrt{\ln 5} - \sqrt{\ln \frac{10}{9}}\right)} \doteq 1.141\,, \tag{49}$$

$$b = \frac{7}{10\left(\sqrt{\ln 5} - \sqrt{\ln \frac{10}{9}}\right)} \doteq 0.741 \tag{50}$$

according to Lemma 2, guarantee that its computations are ε-stable for e.g.

$$\varepsilon = 0.05\,, \tag{51}$$

since they meet

$$|\sigma_G'(\xi)| \leq \left|\sigma_G'\left(\xi_1 + \frac{\varepsilon}{b}\right)\right| \doteq 0.672 < b \tag{52}$$

for all $\xi \in D(\xi_1, \xi_2, \varepsilon) \doteq [0.257, 0.392] \cup [1.201, 1.336]$ as required in (27).

In addition, it follows from (33), (34) that $\|\vec{\varrho}(\vec{z}) - \vec{x}\| \leq \varepsilon$ implies

$$\left|\sigma_G\left(\frac{\|\vec{\varrho}(\vec{z}) - \vec{c}\|}{b}\right) - \sigma_G\left(\frac{\|\vec{x} - \vec{c}\|}{b}\right)\right| \leq \frac{\left|\sigma_G'\left(\xi_1 + \frac{\varepsilon}{b}\right)\right|}{b}\varepsilon < 0.91\varepsilon \tag{53}$$

for any $\vec{z} \in \{0,1\}^n$ and $\vec{x} \in \Re^n$ (cf. (24)). Thus in the course of RBF neuromaton computation, the sequence of absolute errors $\|\vec{\varrho}(\vec{z}^{(t)}) - \vec{y}^{(t)}\|$, $t \geq 0$,

Table 2
Computation of the RBF neuromaton for input $(10)^{20}1$

t	$y_{\mathrm{inp}}^{(t)}$	analog representation of automaton states		t	$y_{\mathrm{inp}}^{(t)}$	analog representation of automaton states	
0	0.2	0.8500000	0.9500000	33	0.9	0.2000230	0.8999770
3	0.9	0.2305155	0.8698093	36	0.2	0.8999902	0.8999876
6	0.2	0.8857692	0.8823592	39	0.9	0.2000053	0.8999947
9	0.9	0.2079568	0.8920347	42	0.2	0.8999978	0.8999972
12	0.2	0.8965211	0.8956301	45	0.9	0.2000012	0.8999988
15	0.9	0.2018928	0.8981089	48	0.2	0.8999995	0.8999993
18	0.2	0.8991912	0.8989748	51	0.9	0.2000003	0.8999997
21	0.9	0.2004371	0.8995628	54	0.2	0.8999999	0.8999999
24	0.2	0.8998140	0.8997641	57	0.9	0.2000001	0.8999999
27	0.9	0.2001004	0.8998996	60	0.2	0.9000000	0.9000000
30	0.2	0.8999573	0.8999459	63		0.2000000	0.9000000

of analog representations $\vec{y}^{(t)}$ of automaton states with binary codes $\vec{z}^{(t)}$ is bounded by geometric sequence $0.91^t\varepsilon$ (precision bounds) converging to zero error. Hence, this RBF neuromaton has even an *error correcting property.* This is illustrated by the recognition of input string $(10)^{20}1$. Thus Table 2 shows the analog representation of the automaton states (and input bits) during the RBF neuromaton computation of length $63 = 21 \cdot 3$ updates. Starting with the initial analog representation (0.85,0.95) of q_0 within precision $\varepsilon = 0.05$ at time $t = 0$, the RBF neuromaton accepts this string with the almost precise final representation (0.2,0.9) of accepting state q_2 at time $t = 63$.

6 Conclusions and Open Problems

In the present paper, the recurrent RBF network with the maximum norm has been exploited for implementing a given deterministic finite automaton within a relatively small network size. For a precise analog representation of the binary values a corresponding simulation works for almost any activation function (Section 3). In addition, for a large class of differentiable activation functions, which are characterized in detail, this implementation proves to be robust with respect to the bounded analog noise in the sense that it preserves a prescribed precision of the underlying representation (Section 4). It follows that the recurrent RBF networks are efficient and reliable analog computational devices that are at least as powerful as the finite automata and thus comparable to perceptron networks [17,28,31].

It would be interesting to generalize the construction of the RBF neuromaton for general norms including the Euclidean one. Another challenge for further research is to prove the upper bounds on the computational power of the analog recurrent RBF networks, e.g. to show that they are able to simulate Turing machines as is known for analog perceptron networks [13,14,16,30].

References

[1] N. Alon, A.K. Dewdney and T.J. Ott, Efficient simulation of finite automata by neural nets, *Journal of the ACM* **38** (2) (1991) 495–514.

[2] J.L. Balcázar, R. Gavaldà and H.T. Siegelmann, Computational power of neural networks: A characterization in terms of Kolmogorov complexity, *IEEE Transactions of Information Theory* **43** (4) (1997) 1175–1183.

[3] D.S. Broomhead and D. Lowe, Multivariable functional interpolation and adaptive networks, *Complex Systems* **2** (3) (1988) 321–355.

[4] R.C. Carrasco, M.L. Forcada, M.Á. Valdés-Muñoz and R.P. Ñeco, Stable encoding of finite-state machines in discrete-time recurrent neural nets with sigmoid units, *Neural Computation* **12** (9) (2000) 2129–2174.

[5] S. Das and M.C. Mozer, A unified gradient-descent/clustering architecture for finite state machine induction, in: J. Cowan, G. Tesauro and J. Alspector, eds., *Neural Information Processing Systems **6**, NIPS'93* (Morgan Kaufmann, San Mateo, 1994) 19–26.

[6] P. Frasconi, M. Gori, M. Maggini and G. Soda, A unified approach for integrating explicit knowledge and learning by example in recurrent networks, in: *Proceedings of the IEEE International Joint Conference on Neural Networks IJCNN'91, Seattle, Vol. **I*** (IEEE Press, New York, 1991) 881–916.

[7] P. Frasconi, M. Gori, M. Maggini and G. Soda, Representation of finite state automata in recurrent radial basis function networks, *Machine Learning* **23** (1) (1996) 5–32.

[8] C.L. Giles, C.B. Miller, D. Chen, H.H. Chen, G.Z. Sun and Y.C. Lee, Learning and extracting finite state automata with second-order recurrent neural networks, *Neural Computation* **4** (3) (1992) 393–405.

[9] M. Gori, M. Maggini and G. Soda, Inductive inference with recurrent radial basis function networks, in: *Proceedings of the Fourth International Conference on Artificial Neural Networks ICANN'94, Sorrento, Italy* (Springer–Verlag, Berlin, 1994), 238–241.

[10] S. Haykin, *Neural Networks: A Comprehensive Foundation* (Prentice-Hall, Upper Saddle River, NJ, 2nd edition, 1999).

[11] B.G. Horne and D.R. Hush, On the node complexity of neural networks, *Neural Networks* **7** (9) (1994) 1413–1426.

[12] B.G. Horne and D.R. Hush, Bounds on the complexity of recurrent neural network implementations of finite state machines, *Neural Networks* **9** (2) (1996) 243–252.

[13] P. Indyk, Optimal simulation of automata by neural nets, in: *Proceedings of the Twelfth Annual Symposium on Theoretical Aspects of Computer Science STACS'95* (LNCS **900**, Springer–Verlag, Berlin, 1995) 337–348.

[14] J. Kilian and H.T. Siegelmann, The dynamic universality of sigmoidal neural networks, *Information and Computation* **128** (1) (1996) 48–56.

[15] S.C. Kleene, Representation of events in nerve nets and finite automata, in: C.E. Shannon and J. McCarthy, eds., *Automata Studies* (Vol. **34** of Annals of Mathematics Studies, Princeton University Press, NJ, 1956) 3–41.

[16] P. Koiran, A family of universal recurrent networks, *Theoretical Computer Science* **168** (2) (1996) 473–480.

[17] W. Maass and P. Orponen, On the effect of analog noise in discrete-time analog computations, *Neural Computation* **10** (5) (1998) 1071–1095.

[18] W. Maass and E.D. Sontag, Analog neural nets with Gaussian or other common noise distribution cannot recognize arbitrary regular languages, *Neural Computation* **11** (3) (1999) 771–782.

[19] P. Manolios and R. Fanelli, First-order recurrent neural networks and deterministic finite state automata, *Neural Computation* **6** (6) (1994) 1155–1173.

[20] M.L. Minsky and S.A. Papert, *Perceptrons* (The MIT Press, Cambridge, MA, 1969).

[21] J.E. Moody and C.J. Darken, Fast learning in networks of locally-tuned processing units, *Neural Computation* **1** (2) (1989) 281–294.

[22] C.W. Omlin and C.L. Giles, Training second-order recurrent neural networks using hints, in: D. Sleeman and P. Edwards, eds., *Proceedings of the Ninth International Conference on Machine Learning* (Morgan Kaufman, San Mateo, CA, 1992) 363–368.

[23] C.W. Omlin and C.L. Giles, Constructing deterministic finite-state automata in recurrent neural networks, *Journal of the ACM* **43** (6) (1996) 937–972.

[24] T. Poggio and F. Girosi, Networks for approximation and learning. in: *Proceedings of the IEEE, Vol. **78*** (IEEE Press, New York, 1990), 1481-1497.

[25] M.J.D. Powell, Radial basis functions for multivariable interpolation: A review, in: J.C. Mason and M.G. Cox, eds., *Proceedings of the IMA Conference on Algorithms for the Approximation of Functions and Data* (Oxford Science Publications, Oxford, 1985) 143–167.

[26] S. Renals, Radial basis function network for speech pattern classification, *Electronics Letters* **25** (7) (1989) 437–439.

[27] C.E. Shannon, The synthesis of two-terminal switching circuits, *The Bell System Technical Journal* **28** (1) (1949) 59–98.

[28] H.T. Siegelmann, Recurrent neural networks and finite automata, *Journal of Computational Intelligence* **12** (4) (1996) 567–574.

[29] H.T. Siegelmann, A. Roitershtein and A. Ben-Hur, Noisy neural networks and generalizations, in: S.A. Solla, T.K. Leen and K.-R. Müller, eds., *Advances in Neural Information Processing Systems **12**, NIPS'99* (The MIT Press, Cambridge, MA, 2000) 335–341.

[30] H.T. Siegelmann and E.D. Sontag, Computational power of neural networks, *Journal of Computer System Science* **50** (1) (1995) 132–150.

[31] J. Šíma, Analog stable simulation of discrete neural networks, *Neural Network World* **7** (6) (1997) 679–686.

[32] J. Šíma and J. Wiedermann, Theory of neuromata, *Journal of the ACM* **45** (1) (1998) 155–178.

[33] J. Šíma and P. Orponen, General-purpose computation with neural networks: A survey of complexity theoretic results, *Neural Computation* **15** (12) (2003) 2727–2778.

[34] M. Šorel and J. Šíma, Robust implementation of finite automata by recurrent RBF networks, in: *Proceedings of the 27th Seminar on Current Trends in Theory and Practice of Informatics SOFSEM'2000, Milovy, Czech Republic* (LNCS **1963**, Springer–Verlag, Berlin, 2000) 431–439.

[35] P. Tiňo and J. Šajda, Learning and extracting initial mealy automata with a modular neural network model, *Neural Computation* **7** (4) (1995) 822–844.

[36] I. Wegener, *The Complexity of Boolean Functions* (Wiley/Teubner, Chichester, 1987).

[37] Z. Zeng, R. Goodman and P. Smyth, Learning finite state machines with self-clustering recurrent networks, *Neural Computation* **5** (6) (1993) 976–990.

Michal Šorel was born in the Czech Republic in 1975. He received the M.Sc. degree in Computer Science at Charles University, Prague, in 1999. He is currently a Ph.D. student in the Department of Image Processing, Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, Prague. His research interests include computer vision, image processing, registration, invariant features in multimodal images, neural networks.

Jiří Šíma received the Ph.D. degree in Computer Science from the Academy of Sciences of the Czech Republic, in 1993. Since 1994 he has been a research fellow in the Department of Theoretical Computer Science, Institute of Computer Science, Academy of Sciences of the Czech Republic, and an Associate Professor at Charles University, Prague, since 2000. He also gives lectures at Masaryk University, Brno. His research interests include complexity theory and neural networks. He is a co-author of a national monograph on the mathematical theory of neural networks.