

# A Synergy of the Rosenblatt Perceptron and the Jordan Recurrence Principle

S. S. Yakovlev and A. N. Borisov

Riga Technical University, 1 Kalku Street, Riga LV-1658, Latvia

e-mail: Arkadijs.Borisovs@cs.rtu.lv, sergeyk@fis.lv

Received July 15, 2008; in final form, November 12, 2008

**Abstract**—The recurrent perceptron RP-1—a combination of the Rosenblatt perceptron and the Jordan network—is proposed. As a result, the perceptron acquires properties of recurrent neural networks, while the learning remains as fast and efficient as in the conventional Rosenblatt model. The recurrent perceptron is tested on the example of an agent education problem, where the “student” is placed into a model environment.

**Key words:** perceptron, recurrent neural network

**DOI:** 10.3103/S0146411609010052

## 1. INTRODUCTION

In the present paper we consider an artificial neural network (ANN)—the recurrent perceptron RP-1, which is based on Rosenblatt’s original investigations of perceptrons (see [1]). In particular we develop some of the ideas outlined by Rosenblatt in the final chapter of his book, which so far have not found practical implementation. At present, widely used algorithms based on error back-propagation (see [2]) face serious performance limitations. As a result, they either cannot be applied to high-dimensional problems, or their application requires a huge expenditure of computing resources.

The famous book by Wasserman [3] had a significant impact on the terminology used in the field of ANNs. Since its publication, the conventional Rosenblatt perceptron is classified as a neural network with a single layer of adjustable connections. It is often pointed out that the classical perceptron is not capable of solving nonlinear problems, for instance, the XOR problem. This is indeed the case if we do not take advantage of an input layer with fixed connection, which actually can map the input data corresponding to a nonlinear problem into a linear representation. After that, the network’s middle layer can successfully solve the problem under consideration. This feature was previously noticed by Bongard [4], who pointed out that the perceptron can map the space of receptors into another space, where the classes are separated in a linear fashion.

The subsequent activity in this field led to the invention of recurrent networks. The most promising of these are networks possessing internal states, such as the Elman network [5] or the Jordan network [6].

Unfortunately, the training process in recurrent networks is associated with an unwarranted complexity of learning algorithms. Among them, we can mention algorithms based on the back-propagation in time, the real-time recurrent learning, and the Kalman filter (see [7]). They all can be considered as advanced versions of the error back-propagation algorithm. In this paper we apply one of the perceptron-type learning algorithms (*learning by error correction*) to recurrent networks. We show that this algorithm, unlike the ones mentioned above, does not suffer from learning slowdown when applied to recurrent networks. We will concentrate our attention only on the learning process, because it is the bottleneck of all neural networks. As far as retrieval is concerned (which is especially important for a network’s forecasting ability), recent investigations show that one can achieve a good level of retrieval quality already in the framework of the Rosenblatt perceptron, without any essential modifications (see [8, 9]).

## 2. CONSTRUCTION OF THE RECURRENT PERCEPTRON RP-1

### 2.1. Recurrent Perceptron Architecture

To turn a perceptron into a recurrent network, we endow the Rosenblatt perceptron with feedback connections and with the context receptors of the Jordan network. However, we are going to use a training pro-

cedure based not on the error back-propagation algorithm, as in the Jordan network, but on the error correction algorithm, as in the Rosenblatt perceptron. Our model also implements many fine novel features, and in order to outline those, we begin with a few definitions specifying the terminology and the functioning mechanism of the proposed recurrent network.

*Definition 1.* An *outside sensor* (S-element) is any sensitive element of a particular detector that transforms the influence of environment (external stimuli) into signals convenient for processing by the network. For instance, a network can receive a flow of data from a video camera, which converts every space point into a single byte of information representing a particular color of 256-color spectrum and passes the signal over a connection with eight binary inputs. The total number of binary inputs obtained from a particular domain of the external environment is determined by the camera's resolution.

*Definition 2.* An *associative element* (A-element) is a threshold device that produces the output signal (+1) when the algebraic sum of its input signals is greater or equal to some threshold value  $\theta > 0$ , otherwise the output signal is absent (zero signal). If the output signal (+1) is present, we say that the A-element is active.

*Definition 3.* A *classifier* (R-element) differs from an A-element only by its role in the ANN. While A-elements, as a rule, occupy an internal middle layer, R-elements pass signals to an output device (or the output signal from R-elements can be read). They can also play the role of signal sources for  $S_{in}$ -elements. R-elements classify stimuli according to a system specified by one of the teachers during training. Therefore, from here on we use the term classifier synonymously with R-element.

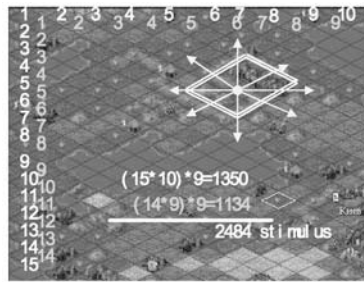
*Definition 4.* The *signal strength* is equal to the value of the weight coefficient (the number of receptors in the postsynaptic neuron) of a particular connection between two elements. We will distinguish excitatory connections, the weights of which are greater than zero, and inhibitory connections, the weights of which are less than zero. The absence of a signal can be interpreted as a signal with zero strength. The strengths of incoming signals are summed up by the A- and R-elements.

*Definition 5.* An *inner sensor* ( $S_{in}$ -element) is a sensitive element of any internal device that registers the internal state of the network appearing during performance of the network.

*Definition 6.* *Sensor modality* is a semantically heterogeneous information arriving at sensor elements. Most often it is visual or acoustic information; however, it can also be a group of related notions (for instance, the number of identical territories or the amount of resources).

*Definition 7.* A *recurrent perceptron* RP-1 is a system which implements the following features:

1. The system is made up of binary S-,  $S_{in}$ -, A-, and R-elements.
2. The system is a perceptron with directed connections joining S-elements (and  $S_{in}$ -elements whose groups are separated over modalities) with A-elements, and A-elements with R-elements.
3. The system includes one-step time-delay feedback connections from R-elements to  $S_{in}$ -elements. These connections allow R-elements to affect the values of  $S_{in}$ -elements during the next time interval.
4. The weights of all connections joining S-elements and  $S_{in}$ -elements with A-elements are fixed (do not change during the training process). A random process with some distribution function chooses them once and for all.
5. The weights of all connections from A-elements to R-elements are adjustable, and they are modified during a training process according to the error-correction algorithm.
6. The number of connections from S-elements and  $S_{in}$ -elements to A-elements is determined by a certain number  $d$ , which is much smaller than the total number of sensor elements, and it is chosen separately for each modality. The total number of input connections of an A-element is equal to the sum of  $d$  numbers over all modalities. The connections from A-elements to R-elements are specified by connection strengths, which are determined during the training process. A certain strength can be equal to zero; in this case, the corresponding connection is effectively absent.
7. The time of internal signal processing is synchronized to a particular frequency  $f$ . The internal processing takes a certain time equal to the length of the stored sequence of stimulus–reaction pairs.
8. As a new external signal with frequency  $f$  is received, the thresholds of all associative elements are set to zero.
9. The learning process can be split into several stages.



**Fig. 1.** Area map where the agent is placed.

## 2.2. Recurrent Perceptron Training

**Definition 8.** The *error-correction procedure* is a training method where the connection strengths are not modified when the perceptron produces a correct response to a particular stimulus. When a wrong response is registered, the relevant connection strengths are changed by a unit amount and the sign (+/-) of the adjustment is opposite to the response error sign.

**Definition 9.** A *training stage* is a segment of the network training process with a particular group of data. Each such group is characterized by the size of the training sample, the training time (measured by the number of required iterations), and by the computer modeling time. The division of the learning process into training stages allows one to pass

information by in certain feasible amounts. At every consecutive stage, we have the network in a partially trained state.

## 2.3. Differences between the Recurrent Perceptron and the Jordan Network

Let us consider how a sequence of signals travels through the recurrent perceptron RP-1. The signals arrive at the group of receptors (external sensors) connecting the perceptron to the outside world, and then they are passed to a hidden layer. The signal transformed by the hidden layer is passed to the output layer (the classifier). Up to this moment, the whole process is virtually the same as in the recurrent Jordan network. However, subsequent signals do *not* leave the network; instead a copy of them is passed to the delay element.

When the signals reach the output layer, the outside sensor does *not* receive another pattern. Instead, the context group of receptors (internal sensor) receives the output from the classifier delayed by the delay element. Thus, only the state of the context layer changes over the last time interval. This, however, is quite sufficient, because, in response, the state of the hidden layer changes as well and the next pattern of a stored sequence is passed to the output layer. The sequence length is determined by the time of internal signal processing, which is synchronized to a particular frequency  $f$ . Only after the internal processing is complete will the whole sequence be read from the output layer and the next pattern passed to the external sensor, triggering the output of a new sequence.

# 3. APPLICATION OF THE RECURRENT PERCEPTRON RP-1: EXAMPLE
















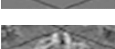
## 3.1. The Agent Education Problem

In artificial intelligence theory there is the notion of a trainable intellectual agent. In our case, this is an artificial intelligence system that observes the model environment and is capable of learning and adapting to a changing environment. In order to actively perform its functions, the agent is usually endowed with a hierarchical structure, which contains many “subagents.” We consider here performance of only the sensor subagent (from here on, simply agent), which creates his/her own “world image” on the basis of signals from its sensors.

The agent uses the recurrent perceptron mechanism, which allows him to react to the information from the model environment and to learn the most important information fragments. In this paper, we do not consider the agent’s behavior from analysis of the obtained information. Instead we concentrate on the learning process itself, the speed of this process, and some other characteristics of it. If we consider the behavior of the trained agent, we should be able to see where the agent moves as he/she investigates the surroundings and where he/she builds a city for extraction of natural resources. The agent would be also able to generate certain rules, which were not given to him externally during the learning process. When placed on a new map, he/she would be also able to determine the best way to behave in an unknown environment. We hope to address these questions in our future investigations. In the present paper, we limit ourselves just to the process of training for the solution of various classification tasks.

The model environment is shown on an area map (see Fig. 1), which is divided into 276 rhombuses (lots) of different types (we use 16 types). Each territory type has different amounts of natural resources, which can be extracted via the process of area development. We distinguish three types of resources. Table 1 shows all territory types that can be found on the map, as well as their characteristics.

## Territory types and their characteristics

The area image	Unique area type identifier* of the territory type	Amount of resources (Resource No. 1/No. 2/No. 3)
	130	2/0/0
	82	1/1/0
	65	1/0/0
	114	1/3/0
	146	2/1/0
	50	0/3/0
	172	2/2/3
	77	1/0/3
	210	3/1/0
	66	1/0/0
	147	2/1/0
	97	1/2/0
	109	1/2/3
	200	3/0/2
	72	1/0/2
	17	0/1/0

\* The unique identifier occupies 1 byte of memory, which allows one to code up to 255 different territory types. In this particular problem, we consider only 16 territory types. However, this is only a particular case; therefore, the network possibilities are not used to their full capacity

During the learning process, the agent goes through all possible positions on the map. At the first stage, the agent walks consecutively through ten external positions in the direction of the  $X$  axis, and through 15 similar positions in the direction of the  $Y$  axis. In total there are 1350 of such possible positions, and their coordinates are marked by white letters in Fig. 1. During the second stage, the agent consecutively goes through nine positions in the direction of the  $X$  axis and over 14 similar positions in the direction of  $Y$  axis. In total there are 1134 of such possible positions; the gray letters in Fig. 1 mark their coordinates.

As the agent passes through the lots of the model environment, he forms a learning sample, which is a collection of the corresponding stimulus–reaction pairs (see Fig. 2). As a stimulus we use the visual image of the nine lots surrounding the agent (a single lot has a size of  $64 \times 32$  pixels with a spectrum of 256 colors) (see Fig. 2: on the left is a magnified detail of the area map). A teacher trains the agent to generate the correct reactions to stimuli. Information presented by the teacher during training, and later required as output, contains the following three components:

1. *The types of the territories surrounding the agent.*

2. *The number of identical territories* of a particular type surrounding the agent. This combination contains 16 numbers—one number for each territory type.

3. *The amount of resources* in the domain investigated by the agent. A resource mask is created this way. It contains three numbers (one number for each resource type) characterizing the total amount of resources in the domain surrounding the agent.

### 3.2. Modeling Results

We developed a program that uses one virtual processor of the Xeon E5310 quad-core processor running under Windows Vista x86. During simulations, the software used 800–950 Mb of RAM from an available 2 Gb. The stored three-component information is represented by three modalities (see Definition 6). Namely, we introduce three groups of  $S_{in}$ -elements (their number and dimensionality are shown at the top-left corner of Fig. 3).


A particular value of parameter  $d$  corresponds to each of the above groups. It specifies the (random) number of connections to a particular A-element (four groups of 20 links). Otherwise, because of the randomness in the number of connections, A-elements may not take into account all the necessary internal modalities. This is particularly important when one of the modalities, like the visual image of the domain in our case, has a noticeably higher dimension than the others. In such a case, the number of connections is distributed according to binomial (or some other, specially created) distribution, and it could happen that no connections would be established with one of the modalities (or the number of established connections would not be sufficient). As a consequence, stimuli, which differ by only one modality, will not be correctly identified by the perceptron, which will result in lack of convergence of the learning process.

Because the number of details in each visual image ( $64 \times 32 \times 8 = 16384 \times 9 = 147456$  bits) is large and distinguishing different lots is not too problematic (we have only 16 different types, plus a certain number of modifications of each region), we can reduce the amount of input data to 5600 bits ( $700 \times 8$ ), so that only a fraction of each particular lot is captured. One cannot use the image of only one lot (for instance, just the central one). In this case, the input information would not be sufficient and the internal processing would not be able to distinguish between cases when the central lot is the same, but the adjacent ones are different. Thus, the volume of input information will be characterized by the number of available external sensors and not by all of the available information.

The number of A-elements depends on the total number of stimuli necessary for memorization (2484). As a rule, it is about 60–80% of the number of necessary stimuli. In our test problem, training is conducted in two stages (1350 + 1134 stimuli) and the perceptron's performance is tested for two cases of network size: 1500 A-elements (the lower boundary) and 2000 A-elements (the upper boundary).

Due to the recurrence of the network, nine internal states are created during its performance. This number is equal to the sequence length, which is produced when the network determines the territory type and computes the number of identical territory types and the amount of resources there. The output layer receives a complete set of information. In the future, such information will help the agent to choose the direction while searching for optimal locations on the area map.

Thus the recurrence makes it possible to perform internal information processing—retrieval of nine standard information sets about each lot, which were stored in the network during the learning stage. More precisely, during each performance cycle, the network not only retrieves a single piece of information about the lot, but also performs certain operations (for instance, summation), which take into account the results of previous stages of information analysis. During the entire period of internal processing, the external sensors contain the same stimulus; nevertheless, because of the nine cycles performed internally, the network produces an output of the required format.



1.	#65	1/0/0	0010-0000-0000-0000
2.	#147	3/1/0	0010-0000-0010-0000
3.	#97	4/3/0	0010-0000-0011-0000
4.	#65	5/3/0	0020-0000-0011-0000
5.	#147	7/4/0	0020-0000-0021-0000
6.	#97	8/6/0	0020-0000-0022-0000
7.	#147	10/7/0	0020-0000-0032-0000
8.	#147	12/8/0	0020-0000-0042-0000
9.	#97	13/10/0	0020-0000-0043-0000

**Fig. 2.** Example of input–output data (stimulus–reaction pairs).

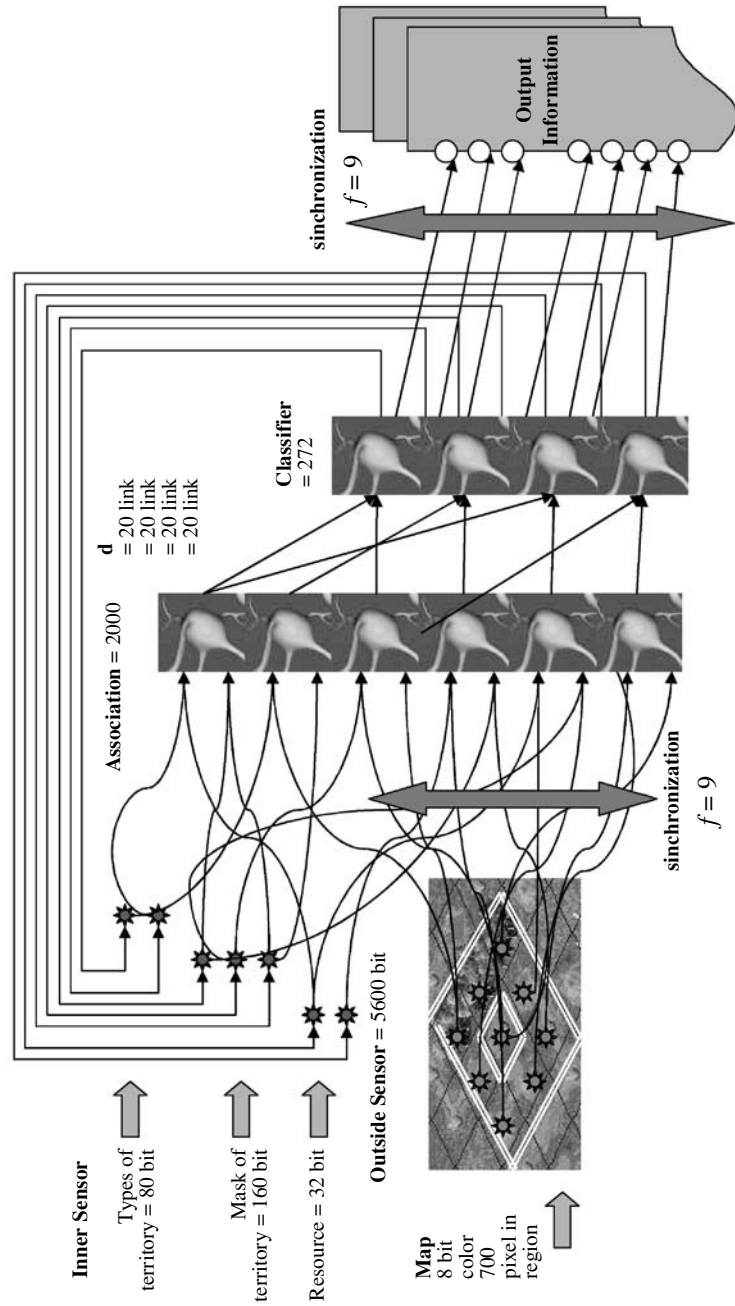
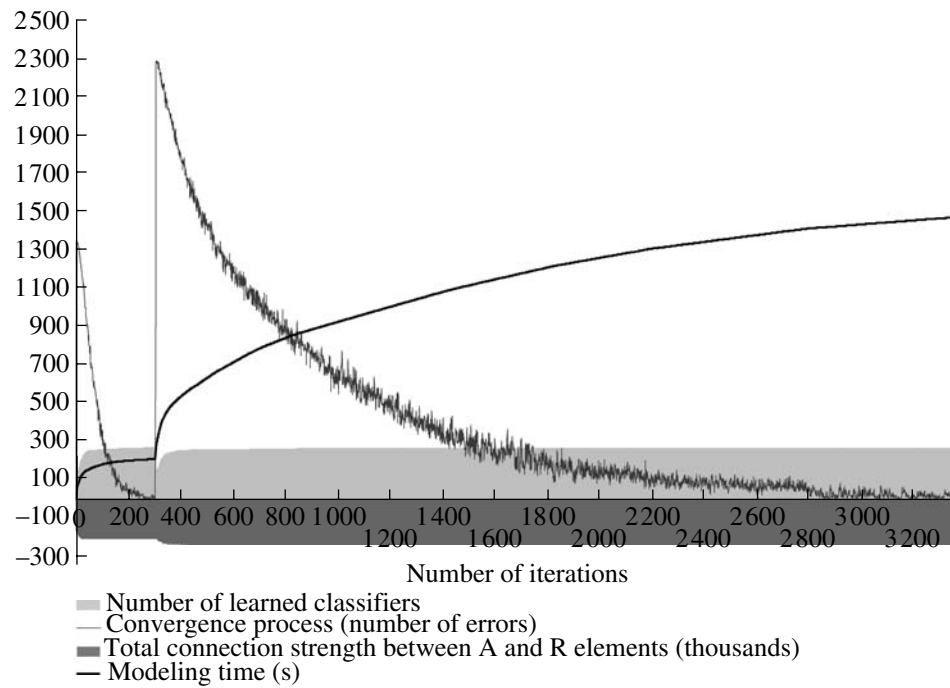
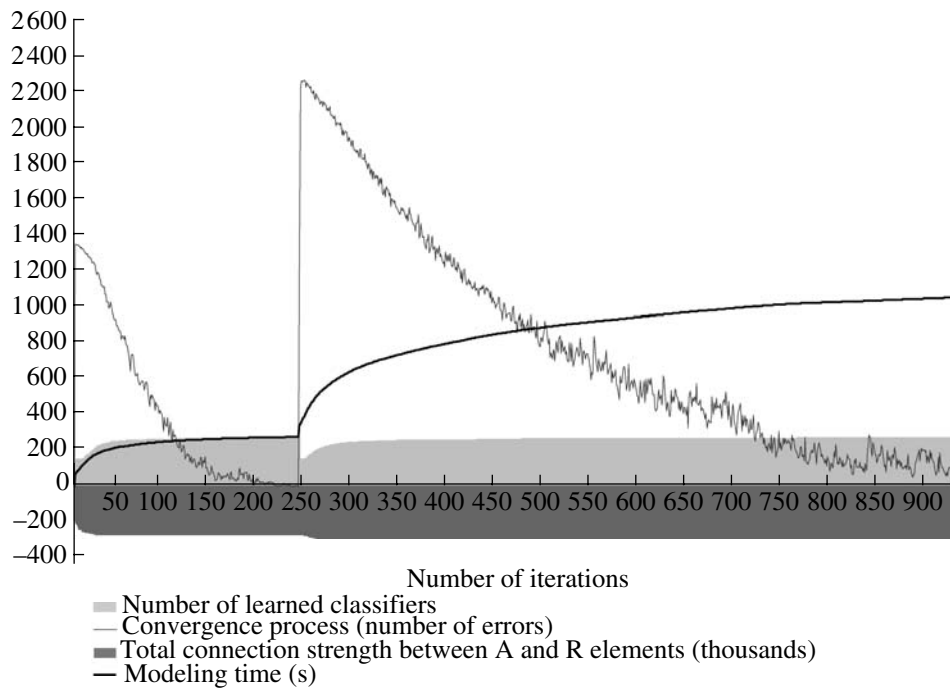


Fig. 3. Recurrent perceptron RP-1 architecture necessary for the considered sensor subagent.



**Fig. 4.** Characteristics of educating of the recurrent perceptron for a minimally necessary (60% of the total number of stimulus) number of A-elements.



**Fig. 5.** The characteristics of educating of the recurrent perceptron for a practically sufficient (80% of the total number of stimulus) number of A-elements.

While experimenting with the network, we kept track of four network parameters. Figures 4 and 5 contain four graphs, which show relative variations of various network characteristics during internal processing cycles.

The  $X$  axis shows the iteration number, while the  $Y$  axis shows the magnitude of a particular network characteristic. For the convergence curve, the  $Y$  axis shows the current number of errors; for the learning-rate curve, the processing time (in seconds) spent up to the current iteration; for the classifiers' learning curve, the number of trained classifiers over the previous cycles; for the variation of connection efficacies, the total connection strength (in thousands) after the previous iteration. Comparing Figs. 4 and 5, we can estimate how all the above characteristics depend on the number of A-elements.

It is necessary to point out that the differences between the first and second learning stages are clearly visible in the graphs (see Definition 9). During the first stage, the network stores half the information. While at the beginning of the second stage, the network forgets nearly all it learned during the first stage. On the convergence curves, the learning stages are marked by two jumps followed by convergence toward zero. However, if we add less information during each stage, then the network will forget less of the previously stored information. If we have more education stages, the whole learning process would take more time. Therefore, it is better to receive information by certain reasonable portions analogous to the existence of short-term memory, which passes its content to long-term memory. Note also that the learning rate during the first stage is greater than that during the second stage. This is because during the first stage, the network has a lot of free memory (the number of A-elements is twice as high as the minimally necessary number) hence the learning rate is very good.

One of the curves shows the number of trained classifiers. Each classifier is trained independently of (in parallel with) the others. And once the training of a particular classifier is finished, that is, once it produces the correct responses for all the training stimuli, we can exclude this classifier from the training program. The time required for training different classifiers varies depending on the complexity of the stimulus-response dependence. As is clearly visible from the corresponding graph, the more classifiers that have already been trained, the less time spent on the next iteration. The learning-rate curve (in seconds) is more sensitive to the number of trained classifiers than to the number of errors.

The dynamics of the total connection strength between A- and R-elements is also of considerable importance. The overall connection strength grows only at first; after that, it remains almost stable until the next portion of information arrives, exhibiting only insignificant fluctuations. It is interesting that the overall connection strength is negative. This means that the number of excitatory connections at each time is small. However, their strength is sufficient for overcoming the background influence of inhibitory connections. That is, at the time the stimulus arrives, we observe a sharp spike in activity. This is the reason for the stability of the learning process. If we have a positive total connection strength, then even a slight stimulus would activate a large number of elements. In such a situation, it would be difficult to differentiate between different stimuli.

#### 4. CONCLUSIONS

We propose recurrent perceptron RP-1, which is capable of forming internal states, of processing cyclically complicated stimuli, and of splitting the processing period into several consecutive segments. Such a recurrent perceptron can be quickly trained using the error correction algorithm. Numerical experiments demonstrate the possibility and the success of applying our recurrent perceptron to the problem of agent control. However, we have concentrated our attention not on the problem of agent control itself or on obtaining answers to such questions as to where it is better to build a city for mineral resource extraction, but on the dynamics of the internal network parameters during the learning process. The analysis of these parameters allows us to understand if the learning process converges and, if yes, what the corresponding convergence rate is irrespective of particular cases. The results in this paper allow us to assert that the proposed recurrent perceptron RP-1 can be applied to solving high-dimensional problems without loss of efficiency in the learning process.

#### ACKNOWLEDGMENTS

The authors are grateful to Professor Aldis Baums for careful reading of the manuscript and constructive remarks.

#### REFERENCES

1. Rosenblatt, F., *Principles of Neurodynamics (Perceptrons and the Theory of Brain Mechanisms)*, Washington D.C.: Spartan Books, 1962.



2. Rumelhart, D.E., Hinton, G.E., and Williams, R.J., Learning Internal Representations by Error Propagation, *Parallel distributed processing: Explorations of the Microstructure of Cognition*, Cambridge: MIT Press, 1986, pp. 318–362.
3. Wasserman, P., *Neurocomputing: Theory and Practice*, New York: Van Nostrand Reinhold, 1990.
4. Bongard, M.M., *The Recognition Problem*, Moscow: Nauka, 1967.
5. Elman, J.L. Finding Structure in time, *Cognitive Science*, 1990, vol. 14, pp. 179–211.
6. Jordan, M.I., *Serial Order: a Parallel Distributed Processing Approach*. Institute of Cognitive Science, University of California, San Diego, 1986, Report 8604.
7. Haykin S. *Neural Networks: A Comprehensive Foundation*, 2nd edition, Upper Saddle River, NJ: Prentice Hall, 1998.
8. Kussul E., Baiduk T., Kasatkina L., Lukovich V., Rosenblatt Perceptrons for Handwritten Digit Recognition, *IEEE 0-7803-7044-9*, pp. 1516–2001.
9. Kussul E., Baiduk T., An Improved Method of Handwritten Digit Recognition Tested on MNIST Database, *Image and Vision Computing*, 2004, vol. 22, pp. 971–981.