Contemporary Engineering Sciences, Vol. 8, 2015, no. 18, 825 - 833
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ces.2015.55160

Capacity of Bidirectional Associative Memory

Kirill Mishchenko

Department of Algebra and Discrete Mathematics
Ural Federal University
620083 Ekaterinburg, Russia

Copyright © 2015 Kirill Mishchenko. This article is distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

The capacity of Bidirectional associative memory (BAM) was examined a lot in research, but not completely. In particular, this issue was not investigated in the context of strings coding. In this paper we apply different approaches to estimate the capacity of BAM for strings coding. One of these approaches is recalling of all coded strings. Another is applying Hamming and Levenshtein distances to recover coded strings. The experiments show poor performance of BAM for the problem of strings coding. In addition, we find a small example of poor performance of BAM. We also discover that there is its extended version. The examples can be used to test advanced methods of coding of strings.

Subject Classification: 68T05

Keywords: Bidirectional associative memory, strings coding, Hamming distance, Levenshtein distance, edit distance

1 Introduction

Bidirectional associative memory (BAM) is a classic architecture of neural networks. It is investigated a lot in research (see e.g. [1, 2]). One of the key issues for BAM researchers is capacity [3]. This issue was examined a lot, but not completely (see e.g. [4, 6]). It is related with existing several ways to estimate BAM's capacity. There are divergent results for different data.

In particular, for a system with n and p units in the first and second layers respectively 90% pairs can be recalled in the presence of $\sqrt{min(n,p)}$ coded pairs [4]. There is the capacity $0.1998n$ on condition that n is equal to p [5]. In [6] an example of storing more than $\sqrt{min(n,p)}$ pairs was given. In our paper we continue to examine capacity of BAM. It should be noted that there are different modifications of BAM [3]. We use the original variant from [7].

2 Notations and definitions

Let training data be defined as a set of m pairs

$$\{[X_i, Y_i] | 1 \leq i \leq m\}, \tag{1}$$

where $X_i = (x_{i1}, \ldots, x_{in})$, $Y_i = (y_{i1}, \ldots, y_{ip})$, $x_{ij}, y_{ik} \in \{-1, 1\}$, $i = 1..m$, $j = 1..n$, $k = 1..p$. In according with [7] a matrix memory M is defined as $M = \sum_{i=1}^{m} X_i^T Y_i$.

It is shown in [7] that an associative vector can be found by transferring neuron's signals with matrixes M and M^T. For example, to find the associative vector for the vector X^0 following operations should be applied:

$$\begin{aligned} Y^0 = f(X^0 M) \rightarrow X^1 = & f(Y^0 M^T) \rightarrow \\ Y^1 = & f(X^1 M) \ \rightarrow ... \rightarrow Y^k = f(X^k M). \end{aligned} \tag{2}$$

As a consequence, stabilized vectors X^k and Y^k are formed. Similarly, the associative vector can be found for the given Y^0 with following operations:

$$\begin{aligned} X^0 = f(Y^0 M^T) \rightarrow Y^1 = & f(X^0 M) \ \rightarrow \\ X^1 = & f(Y^1 M^T) \rightarrow ... \rightarrow Y^k = f(Y^k M^T). \end{aligned} \tag{3}$$

For any given pair (X^i, Y^i) the value of the potential function E can be matched [7]. The value is defined as

$$E(X, Y) = -XMY^T. \tag{4}$$

In [7], it was proved that every step in (2) and (3) leads to decreasing corresponding values of the potential function (4) up to a local minimum at (X^k, Y^k) in according with previous notations.

Let a training set (1) be given. Let $X^0 = (x_1^0, \ldots, x_n^0)$, $x_i^0 \in \{-1, 1\}$, $i = 1..n$. Using the operations (2) for X^0 we find $Y^k = (y_1^k, \ldots, y_p^k)$, $y_j^k \in \{-1, 1\}$, $j = 1..p$, $X^k = (x_1^k, \ldots, x_n^k)$, $x_i^k \in \{-1, 1\}$, $i = 1..n$, such that (X^k, Y^k) is a local minimum of (4). Using the operations (3) for Y^k we next find $\mathfrak{Y}^l = (\mathfrak{y}_1^l, \ldots, \mathfrak{y}_p^l)$, $\mathfrak{y}_j^l \in \{-1, 1\}$, $j = 1..p$, $\mathfrak{X}^l = (\mathfrak{x}_1^l, \ldots, \mathfrak{x}_n^l)$, $\mathfrak{x}_i^l \in \{-1, 1\}$, $i = 1..n$, such that $(\mathfrak{X}^l, \mathfrak{Y}^l)$ is a local minimum of (4). Then the vector $\mathfrak{X}^l$ will be called the *preimage* for the vector X^0.

3 BAM and string's effective representation

Software agents usually should be background and efficient enough for using computational resources to not affect user's tasks performance. When the environment of intelligent agents is a file system there is a need of processing large text data consisting of paths of files. Intelligent systems based on widely used methods such as back-propagation neural networks are typically much less efficient in the case of large data [8]. Thus, the issue of string's effective representation appears. This issue consists in with finding such mapping of text information which allows to reduce the dimension of data.

Examining of BAM's applicability for the mentioned issue is mainly related with wide using this kind of neural networks as a classic tool allowing to code pairs of vectors and recall one with another. Another reason of choosing BAM is a simple implementation. Consequently BAM is a naturally chosen model for using in issues such as string's effective representation while developing various agents. On the other hand, BAM's capacity in the context of such issues has not not been investigated. Thus string data used by agents can be applied as a text set allowing to obtain additional information about BAM's capacity.

Now we consider how to apply BAM to coding of strings consisting of ASCII characters. For the purpose of using ASCII strings as a training set for BAM they can be converted to binary sequences by using ASCII character's codes. In the case when strings have different lengths, binary views of strings which have less lengths can be expanded by adding zeros. After that binary views of all strings should have the same length. The last step is to change 0 in binary views of strings by -1 for getting bipolar views. Obtained bipolar views can be used as left or right elements of pairs in (1).

For the purpose of getting the original string from a bipolar view the inverse transformations should be done in back order. It should be noted that while converting a binary view to the character's sequence bits are joined to bytes, and last zero bytes are eliminated. Retain bytes are considered as appropriate ASCII characters.

Let a training set (1) be given. Let left elements of its pairs are obtained from ASCII strings. Let an ASCII string s be given. Let $X^0 = (x_1^0, \ldots, x_n^0)$, $x_i^0 \in \{-1, 1\}$, $i = 1..n$, be a bipolar vector for the string s. If $\mathfrak{X}^l = (\mathfrak{x}_1^l, \ldots, \mathfrak{x}_n^l)$, $\mathfrak{x}_i^l \in \{-1, 1\}$, $i = 1..n$, is a preimage for X^0, then the ASCII string gotten from $\mathfrak{X}^l$ will be called the *preimage-string* for the string s.

In our experiments training sets consist of strings which are file paths. Examples of such strings are shown on the figure 1. These strings are paired with string containing random visible ASCII characters. For generating visible ASCII characters the random generator of the standard C++ library is used. Each file path contains no less than 26 characters.

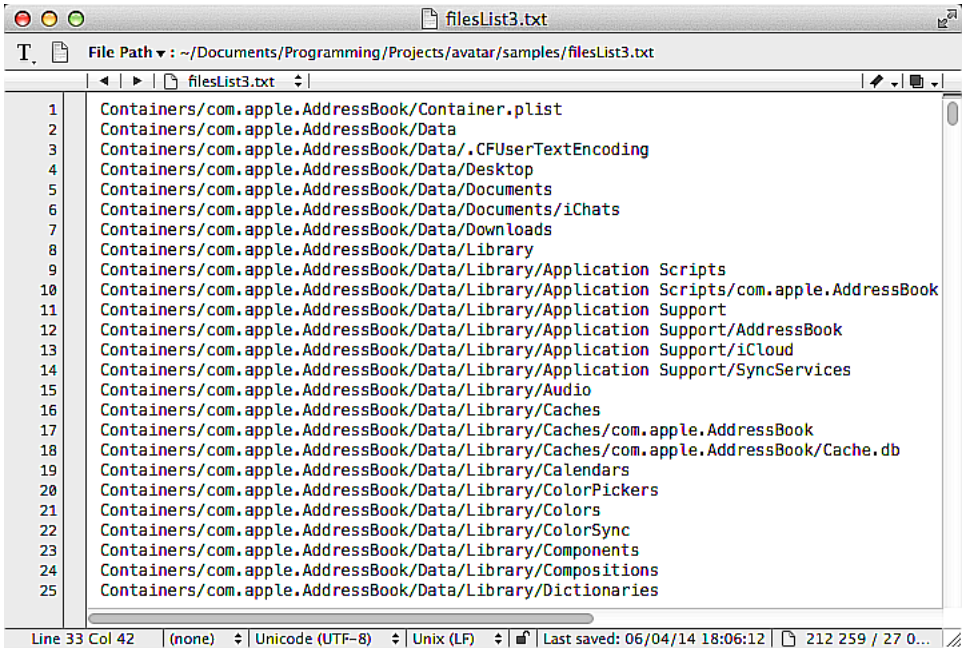


Figure 1: Some of file paths used for making training sets

There are 200 training sets. In each training set the number of pairs m is equal to 10. Numbers of units n and p meet following conditions: $n \geq 26 \cdot 8 = 208$, $p = 26 \cdot 8 = 208$.

There are several ways to evaluate capacity of BAM. The reason of it is not only different kinds of experiments, but also existing a few values to estimate. In particular, one of them is a number of recalled pairs.

In this subsection we consider the method offered in [4]. The first part of the method consists in generating 200 training sets for BAM. For each of training sets, a number of pairs m meets $2 \leq m \leq 16$; numbers of units n and p meet $4 \leq n \leq 256$ and $4 \leq p \leq 256$ accordingly. The vectors are bipolar. Also each of the training sets meets the condition

$$m \leq \sqrt{min(n, p)}. \tag{5}$$

Elements of vectors of all trainings sets are automatically generated by using the Microsoft C embedded random generator.

A set of data is called an event. If all pairs in that set can be recalled, it is declared a success. Otherwise, it is declared a failure. The percentage of success obtained in the experiment described in [4], is 90%.

We carry out an analogous experiment for training sets consisting of strings representing file paths. For each training set BAM is made. Next, for each string a preimage-string is found. If a string and its preimage-string are the same, then it is considered as successful recalling.

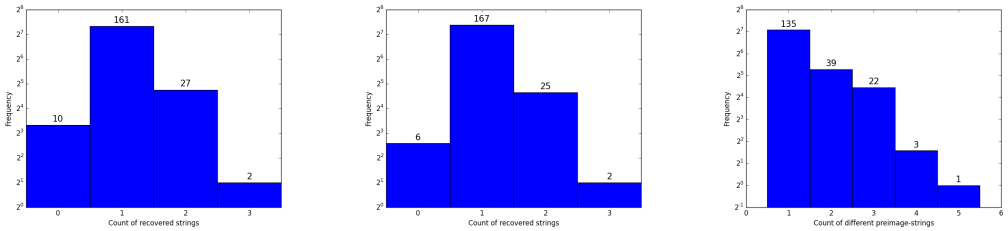
The Hamming distance between two vectors of the same length is defined as the number of coordinates for which the vectors are different. For example, for vectors $(0,1,1,0,0,1,0,0)$, $(0,1,1,0,0,0,0,1)$ the Hamming distance is two. As strings can be represented as vectors, the Hamming distance can be calculated for strings as well. The Levenshtein distance between two strings is defined as the minimum number of operations needed to transform the first string into the second [9]. The operations are insertions, deletions and substitutions [9]. The Levenshtein distance is also referred as edit distance [9]. In [9] the algorithm of finding the Levenshtein distance is given. Let two strings S_1 and S_2 be given. Let $D(i,j)$ denotes the Levenshtein distance between two strings, where one of them is composed of the first i characters of S_1, another is composed of the first j characters of S_2. Using this notation, if S_1 and S_2 have m and n letters accordingly, then the Levenshtein distance between S_1 and S_2 is $D(m,n)$. Also let $S(i)$ denotes the character of S with index i. Indexes start from 1. In according with [9] the value of $D(i,j)$ can be calculated by the rule $D(i,j) = min\{D(i-1,j)+1, D(i,j-1)+1, D(i-1,j-1)+t(i,j)\}$ where $D(i,0) = i$, $D(0,j) = j$, $t(i,j) \in \{0,1\}$, and $t(i,j) = 1$ if and only if $S_1(i) = S_2(i)$. We execute experiments for strings from training data and their preimage-strings. While calculating Hamming and Levenshtein distances we use binary views of strings. While calculating Hamming distances in the cases, when a string and its preimage-string have different lengths, the binary view of one of them which has less length is extended by adding zeros.

Let a training set (1) be given and left elements of its pairs are obtained from ASCII strings $\{s_1, s_2, ..., s_m\}$. Let s'_i be a preimage-string for s_i. Let $d(a,b)$ denotes either Hamming and Levenshtein distances for strings a and b. We say s_i is recovered with a distance d, if $d(s_i, s'_i) < d(s_j, s'_i)$ for all j such that $1 \le j \le m, j \ne i$.

In the experiment for recalling of all strings we obtain 11 BAMs among 200 have just one recalled string. Others have none. Thus the experiment shows that the percentage of success is 0.

As the result of 200 experiments there is the following distribution of counts of recovered strings in a single training set with the Hamming distance: recovering of no strings constitutes 5% of all cases; recovering of one string constitutes 80.5% of all cases; recovering of two strings constitutes 13.5% of all cases; recovering of three strings constitutes 1% of all cases. There are similar results for the Levenshtein distance. They are the following: recovering of no strings constitutes 3% of all cases; recovering of one string constitutes 83.5% of all cases; recovering of two strings constitutes 12.5% of all cases; recovering of three strings constitutes 1% of all cases. The corresponding histograms are shown on the figures 2(a) and 2(b).

During experiments it is noted that there is a small amount of different preimage-strings. The case of one unique preimage-string makes up 67.5%



(a) (b) (c)

Figure 2: The histograms showing results of experiments. (a) and (b) depict the distributions of counts of recovered strings in a single training set with the Hamming distance and with the Levenshtein distance accordingly. (c) depicts the distribution of counts of different preimage-strings

of all cases. The case of two unique preimage-strings makes up 19.5%. The case of three unique preimage-strings makes up 11%. The case of five unique preimage-strings makes up 0.5%. The corresponding histogram is shown on the figure 2(c).

To receive good results of recovering with Hamming and Levenshtein distances there need to be different preimage-strings for testing strings, while in our experiments this condition wasn't met.

4 Searching difficult data

There has been presented results of experiments showing impossibility of recovering strings coded with BAM. To increase confidence in obtained results it is reasonable to find a small example showing the same behavior of BAM. Such example can be used for testing advanced methods of coding of strings to exhibit their effectiveness.

In [6] the right part of (5) was mentioned as a BAM capacity measure used in a literature. Furthermore, in [10] it was said that there was a good enough quality of recovering in the case of meeting (5). Let consider an example meeting (5) but showing failing to recover.

Let following vectors be defined as:

$$
\begin{aligned}
X_1 &= (\ 1\ 1\ \ 1\ \ 1\ -1\ 1\ 1\ -1\ -1\ -1\ -1\ -1\ 1\ 1\ -1\ -1),\\
Y_1 &= (-1\ 1\ -1\ -1\ \ 1\ 1\ 1\ -1\ -1\ -1\ -1\ -1\ 1\ 1\ -1\ -1),\\
X_2 &= (\ 1\ 1\ \ 1\ \ 1\ -1\ 1\ 1\ -1\ \ 1\ -1\ -1\ -1\ 1\ 1\ -1\ -1),\\
Y_2 &= (-1\ 1\ -1\ -1\ \ 1\ 1\ 1\ -1\ \ 1\ -1\ -1\ -1\ 1\ 1\ -1\ -1),\\
X_3 &= (\ 1\ 1\ \ 1\ \ 1\ -1\ 1\ 1\ -1\ -1\ \ 1\ -1\ -1\ 1\ 1\ -1\ -1),\\
Y_3 &= (-1\ 1\ -1\ -1\ \ 1\ 1\ 1\ -1\ -1\ \ 1\ -1\ -1\ 1\ 1\ -1\ -1).
\end{aligned}
$$

The vectors X_1, X_2, X_3, Y_1, Y_2, Y_3 are bipolar views of strings "o1", "o2", "o3", "r1", "r2", "r3" accordingly.

For the training set $\{[X_1, Y_1], [X_2, Y_2], [X_3, Y_3]\}$ let calculate the matrix memory M in according with $M = \sum_{i=1}^{m} X_i^T Y_i$ as

$$M = \sum_{i=1}^{3} X_i^T Y_i =$$

$$\begin{pmatrix}
-3 & 3 & -3 & -3 & 3 & 3 & 3 & -3 & -1 & -1 & -3 & -3 & 3 & 3 & -3 & -3 \\
-3 & 3 & -3 & -3 & 3 & 3 & 3 & -3 & -1 & -1 & -3 & -3 & 3 & 3 & -3 & -3 \\
-3 & 3 & -3 & -3 & 3 & 3 & 3 & -3 & -1 & -1 & -3 & -3 & 3 & 3 & -3 & -3 \\
-3 & 3 & -3 & -3 & 3 & 3 & 3 & -3 & -1 & -1 & -3 & -3 & 3 & 3 & -3 & -3 \\
3 & -3 & 3 & 3 & -3 & -3 & -3 & 3 & 1 & 1 & 3 & 3 & -3 & -3 & 3 & 3 \\
-3 & 3 & -3 & -3 & 3 & 3 & 3 & -3 & -1 & -1 & -3 & -3 & 3 & 3 & -3 & -3 \\
-3 & 3 & -3 & -3 & 3 & 3 & 3 & -3 & -1 & -1 & -3 & -3 & 3 & 3 & -3 & -3 \\
3 & -3 & 3 & 3 & -3 & -3 & -3 & 3 & 1 & 1 & 3 & 3 & -3 & -3 & 3 & 3 \\
1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & 3 & -1 & 1 & 1 & -1 & -1 & 1 & 1 \\
1 & -1 & 1 & 1 & -1 & -1 & -1 & 1 & -1 & 3 & 1 & 1 & -1 & -1 & 1 & 1 \\
3 & -3 & 3 & 3 & -3 & -3 & -3 & 3 & 1 & 1 & 3 & 3 & -3 & -3 & 3 & 3 \\
3 & -3 & 3 & 3 & -3 & -3 & -3 & 3 & 1 & 1 & 3 & 3 & -3 & -3 & 3 & 3 \\
-3 & 3 & -3 & -3 & 3 & 3 & 3 & -3 & -1 & -1 & -3 & -3 & 3 & 3 & -3 & -3 \\
-3 & 3 & -3 & -3 & 3 & 3 & 3 & -3 & -1 & -1 & -3 & -3 & 3 & 3 & -3 & -3 \\
3 & -3 & 3 & 3 & -3 & -3 & -3 & 3 & 1 & 1 & 3 & 3 & -3 & -3 & 3 & 3 \\
3 & -3 & 3 & 3 & -3 & -3 & -3 & 3 & 1 & 1 & 3 & 3 & -3 & -3 & 3 & 3
\end{pmatrix}.$$

In according with (2) the associative vector for X_1 can be calculated as

$$Y_1' = X_1 * M =$$

$$\begin{pmatrix} -42 & 42 & -42 & -42 & 42 & 42 & 42 & -42 & -10 & -18 & -42 & -42 & 42 & 42 & -42 & -42 \end{pmatrix}.$$

If we replace elements of Y_1' by values of *signum*, we get

$$Y_1'' = \begin{pmatrix} -1 & 1 & -1 & -1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & -1 & -1 \end{pmatrix}.$$

It is seen that Y_1'' is not equal to Y_1 but is equal to Y_0.

In the similar way it can be shown that the associative vector for X_2 is not equal to Y_2 but is equal to Y_0

Thus we have found a small example of a training set of BAM meeting the condition (5), but for which not all vectors from the training set are recoverable. The small example of training data confirms poor performance of BAM for the problem of strings coding. Nevertheless, to increase confidence more in obtained results, to test advanced methods of coding of strings and to discover the issue to a greater extent, it can be useful to find a more general example

of training data for BAM meeting the condition (5), but leading to failing to recover some pairs of vectors.

Let the symbol $*$ denotes the operation of concatenation of strings, for example "str1" $*$ "str2" = "str1str2". Let ("c")^k denotes the string consisting of k characters "c". Let $a_i^{(k)} =$ ("o")$^k *$ "i", $b_i^{(k)} =$ ("r")$^k *$ "i". Let $X_i^{(k)}$ and $Y_i^{(k)}$ denotes bipolar views of strings $a_i^{(k)}$ and $b_i^{(k)}$ accordingly. Thus a set

$$\{(X_1^{(k)}, Y_1^{(k)}), (X_2^{(k)}, Y_2^{(k)}), (X_3^{(k)}, Y_3^{(k)})\} \tag{6}$$

is a training set for every positive integer k.

With a computational experiment the ability to recover vectors coded by BAM was checked. The experiment carried out for integer k from 2 to 1000. The results were the same as for the small example: two pairs of three failed to recover. Based on these results it is logical to propose the hypothesis that the similar behavior will take place for every positive integer k.

Obtained results extend knowledge about the capacity of BAM gotten by other researchers. In particular, in addition to the estimate (5) in [5] and [**?**] the authors gave the estimate $0.1998n$ in the case when numbers of units in both layers are equal. This estimate allows to have a finite number of failings to recover. It means that every BAM built upon $0.1998n$ pairs of vectors and having the same numbers of units in both layers, have to allow to recover all pairs except finite number as n approaches infinity.

5 Conclusion

In this paper the possibility to code strings with BAM was examined. Experiments showed that meeting condition (5) used in a literature does not guarantee a good quality of recovering coded strings. In the cases when strings are recovered partially, it is possible to try to identify strings with Hamming and Levenshtein distances. However, to receive good results of recovering with Hamming and Levenshtein distances there need to be different preimage-strings for testing strings, while in our experiments this condition was not met.

We found a small example of a training set of BAM meeting the condition (5), but for which not all vectors from the training set are recoverable. In addition here was found a more general example showing the same behavior. These examples increase confidence in poor performance of BAM for the problem of strings coding and can be used to test advanced methods of coding of strings.

In the future the analytical proofing of the general example can be done. It is also interesting to try to find another structures of examples of failing to recover strings.

References

[1] S. Chartier, C. Leth-Steensen, and M.-F. Hébert, Performing complex associations using a generalised bidirectional associative memory, *Journal of Experimental & Theoretical Artificial Intelligence*, **24** (2012), 23-42. http://dx.doi.org/10.1080/0952813x.2010.535712

[2] S.D. Heon, O.A. Suk, and W.Y. Woon, Activity centered design of smart phone user interface: Learning app execution patterns with neural network model, *International Journal of Smart Home*, **8** (2014), 101-106.

[3] M. E. Acevedo-Mosqueda, C. Márquez, and M. Acevedo-Mosqueda, Bidirectional associative memories: Different approaches, *ACM Computing Surveys*, **45** (2013), 1-30. http://dx.doi.org/10.1145/2431211.2431217

[4] Y.-F. Wang, J. B. Cruz Jr., and J.H. Mulligan Jr., Two coding strategies for bidirectional associative memory, *IEEE Transactions on Neural Networks*, **1** (1990), 81-92. http://dx.doi.org/10.1109/72.80207

[5] H. Shouno, S. Kido, and M. Okada, Analysis of bidirectional associative memory using self-consistent signal to noise analysis and statistical neurodynamics, *Journal of the Physical Society of Japan*, **73** (2004), 2406-2412. http://dx.doi.org/10.1143/jpsj.73.2406

[6] Y. P. Singh, V.S.Yadav, A. Gupta, and A. Khare, Bi directional associative memory neural network method in the character recognition, *Journal of Theoretical and Applied Information Technology*, **5** (2009), 382-386.

[7] B. Kosko, Bidirectional associative memories, *IEEE Transactions on Systems, Man and Cybernetics*, **18** (1988), 49-60. http://dx.doi.org/10.1109/21.87054

[8] G. Jananii, Applying multi layer feed forward neural networks on large scale data, *Bonfring International Journal of Man Machine Interface*, **2** (2012), 1-3.

[9] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational biology*, Cambridge University Press, New York, 1997. http://dx.doi.org/10.1017/cbo9780511574931

[10] S. Osowski, Sieci neuronowe do przetwarzania informacji, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa, 2000. (in Polish)

Received: June 7, 2015; Published: September 9, 2015