Implementing Knowledge Database in Neural Networks

Rafał Bogacz
Computer Science and Management Department
Technical University of Wrocław

Abstract:
Knowledge database coded as triples: Agent, Relationship, Patient may be easily implemented in neural network. In the described program all the arguments of relations are coded in patterns - orthogonal sequences of bits. After giving patterns of Agent and Relationship to the input of the network, there is a pattern of the proper Patient in the output of the network. BAM network was used in the system. The learning rule of this network is so easy, that controlling of the network can be done also by a neural network. In the described problem BAM network obtained better results, when signal was being sent between the layers only one time. The possibility of using Hopfield network to filter noises introduced by the layer network has been discussed.

1. Introduction

There are two approaches to the artificial intelligence: based on logic expert systems and imitating working of human neurons neural networks. The aim of my work is to combine the two approaches - to create expert system, which would be one big neural network or consist of a few cooperating networks. In this article one describes one part of this problem - neural implementation of the part of expert system - knowledge database.

One of the possibilities of representing knowledge is remembering relations between objects [4]. For example: the sentence "Filemon likes milk" can be remembered as relation of liking (called further Relationship, designated R), elements of which are "Filemon" - the "agent" of this relation (called further Agent, designed A) and milk - "patient " of relation (called further Patient, designated P). We know usually Agent and Relationship and we ask about Patient - in this case it would form such a question:

"What does Filemon like?"

The aim of this work is creating the system which is the neural network, remembering these relations.

2. Theoretical project of system

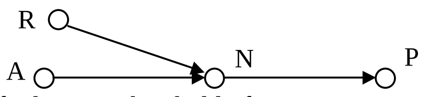
The described program is conversation system. Entering from command line information has a very simple syntax:

<Agent> <Relationship> <Patient> - sentence to remember, e.g. "Filemon likes milk".

? <Relationship> <Agent> - question about Patient, e.g. "? likes Filemon". After asking this question system should answer: "milk". (1)

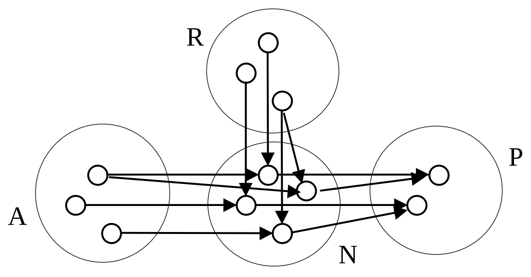
2.1. Representation of relations in neural networks

In the local representation model [1] each object is represented by one neuron. As object one should understand any component of relation (A, R or P). In order that network would satisfy the formed earlier assumptions (1), for each relation, after activation of neurons which represent A and R, representing P neuron should activate. One object can be a Patient in many relations. Therefore it is difficult to find weights of direct connections between neurons, in order to satisfy the assumptions (1). To fulfil the assumptions it is enough to add one neuron for each relation to network.

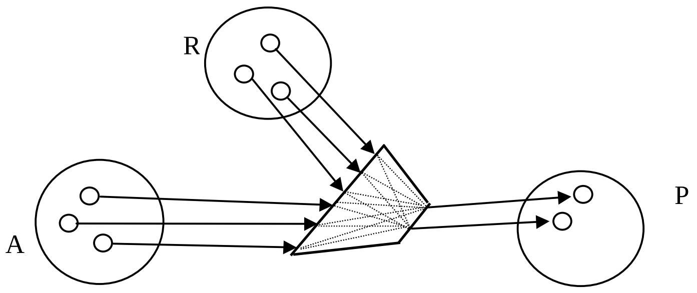


It is enough to find proper threshold of neuron N, to see the network satisfies the assumptions - after activating neurons A and R (and only then) neuron P is activated (and no other neuron).

It is even better visible, when the whole set of relations is coded in this way (to simplify one should assume that sets A, R, P are disjoint) .

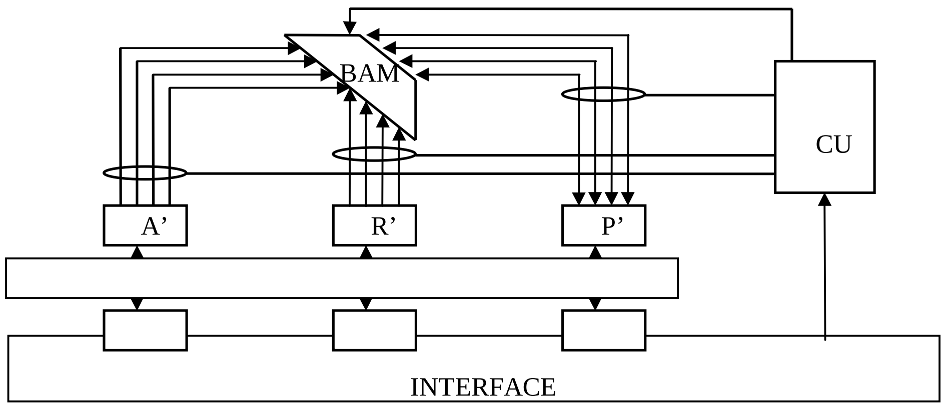


In distributed representation model [1], each object is not represented by one neuron, but by the pattern of activation of lots of neurons. This sequence of values will be further called the pattern. So each object can be represented not by the neuron, but by the pattern. One can also replace additional neurons N, by layer network. Its inputs are the signals from neurons A and R and its outputs activate neurons P. This network should be trained in such a way, that if one gives patterns A and R of the relation, the network would give us in output the proper pattern P.



2.2. Project of system

The assumption of this work was, that the whole system would consist of only cooperating with one another neural networks. Also controlling of system work can be done by neural network, but than learning of layer network must be very easy. Therefore BAM [2] network is used in the system, learning of which is based on extremely easy Hebb learning rule. That is the model of this kind of system



Interface reads commands from command line and in the dependence of the syntax (point 2) it writes proper words to registers A, R, P, and gives information to control unit.

Words entered form keyboard have to be coded to create the pattern. The pattern for each entered word is created as a sequence of bits of settled length (this length will be designated further as DL). The bits can have values {-1,1}.

The pattern is generated at random, but in such a way that these patterns would be orthogonal, i.e. that for each pair of different patterns a, b value:

$$\sum_{i=1}^{DL} a_i b_i \qquad (2)$$

would be minimal, and if it is possible, equal to zero. Orthogonal patterns give bigger capacity of neural network.

Identification unit creates patterns for new entered words, translates words to the patterns and the patterns to the words.

The neural network BAM [2] is the heart of the whole system. This network has 2*DL neurons in the input layer and DL neurons in the output layer. In the input of network is provided concatenation of patterns A and R of the relation, and to the output of network pattern P. The weights of this network are settled by the equation:

$$w_{ij} = \sum_{k=1}^{l} y_i^k x_j^k \qquad (3)$$

where l - number of patterns;

x_j^k - value of j bit of k learnt input (input is concatenation of patterns A, R of k relation).

y_i^k - value of i bit of k learnt output (output is the pattern P of k relation).

In order to teach the network a new relation, it is enough to modify weights in a very simple way:

$$w'_{ij} = w_{ij} + y_i^{k+1} x_j^{k+1} \qquad (4)$$

In this model learning is very simple - it is enough to enter triple of patterns R, A, P from registers to proper neurons, and then insert the network to “the learning state”, in which the weights of connections change proportionally to values of neurons activating, which they connect.

For the above algorithm the control unit of the whole network could be constructed only of a few gates, so implementing CU in neural network also would not be any problem.

3. Software system implementation

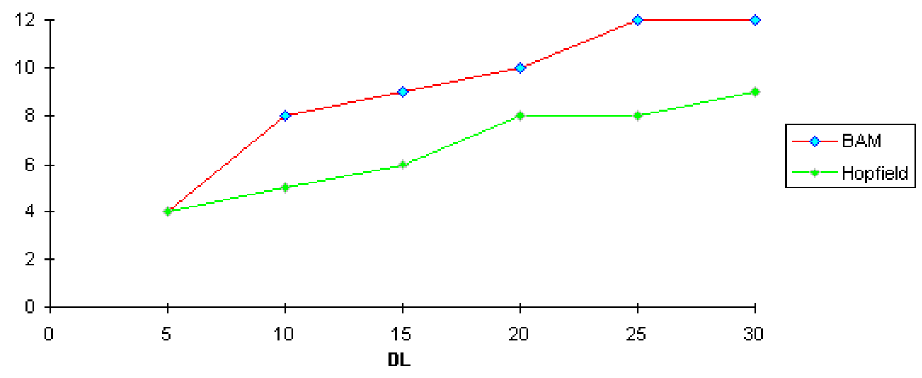
The controlling is not done by neural network in software system implementation. One concentrated the implementation of layer network.

One also improved learning rule. At the beginning the network learns at one time each relation and one modifies the weight according to equation 4. If the network does not remember all the relations (i.e. in some cases gives wrong results) it learns relations, which it did not remember, till it remembers all of them, or after 10 cycles of presenting relations one stops learning.

3.1. Capacity of the network

Number of relations, which network manages to learn for the settled number of neurons is a random variable. The number was approximated for the settled value DL in the way "try 3 times". One considers that network can learn the number of relations, if it succeeds in one of 3 attempts. The number of relations, which network succeeds to learn is shown at chart 1 (series BAM).

Chart 1 - Number of remembered relations in BAM and Hopfield networks

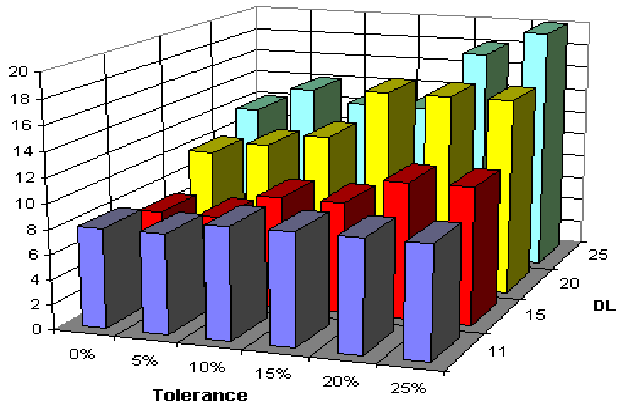


As literature says [3], the recurrent networks are able to remember almost all orthogonal patterns. The number of orthogonal patterns, which can be coded on DL bits, is DL. In this model 3 patterns fall to one relation. Therefore for higher DL, the number of remembered relations approaches DL/3.

3.2. Tolerance of errors

In the system the entered words are coded to orthogonal patterns. The Hamming distance of orthogonal patterns with length DL is DL/2. Therefore, if the output of network was wrong only in a few bits, information is still univocal. It concerns specially high DL. The number of relations, which network can learn is much higher, when one tolerates some percent of wrong answers in the output layer. It is shown by chart 2. By tolerance the acceptable percent of output neurons, which give wrong answers, is designated.

Chart 2 - Number of remembered relations by neural network with error tolerance

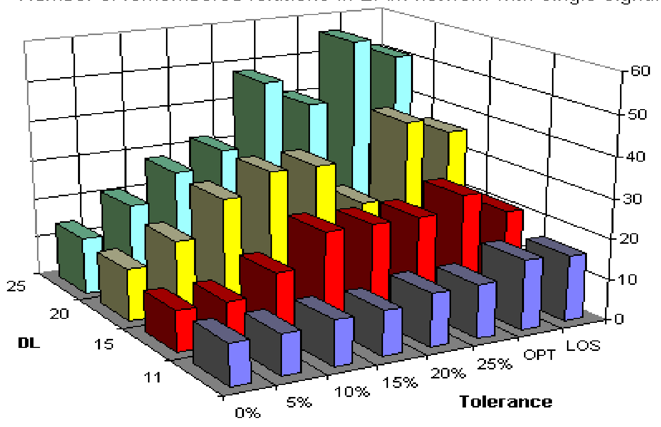


3.3. BAM network with single signal sending

If network learnt the given relation, so after giving into the input layer the patterns A and R of the relation, then after the first signal sending, there is a proper P pattern in the output layer and the network is stable. If network learnt the relation not exactly (i.e. after giving to the input layer patterns A and R, the pattern in the output layer is different on a few bits than pattern P), one can observe the interesting thing. When we give to the input layer learnt pattern (concatenation A and R), after the first sending from input layer to output, there is the pattern in output layer which is different than learnt (P) in a few bits. During further iterations the differences between pattern in output layer and learnt pattern (P) are bigger and bigger, so the error grows. In this case in BAM network the multiple sending signals between layers is not necessary. If the network stabilizes on learnt pattern, it does it in the first iteration. If the network does not find the right pattern in the first iteration, it will never do it.

The BAM provides with better results, in which the signal is sent only once from input layer to output, and not till the stabilization of the network. Results are shown in chart 3.

Chart 3 - Number of remembered relations in BAM network with single signal sending

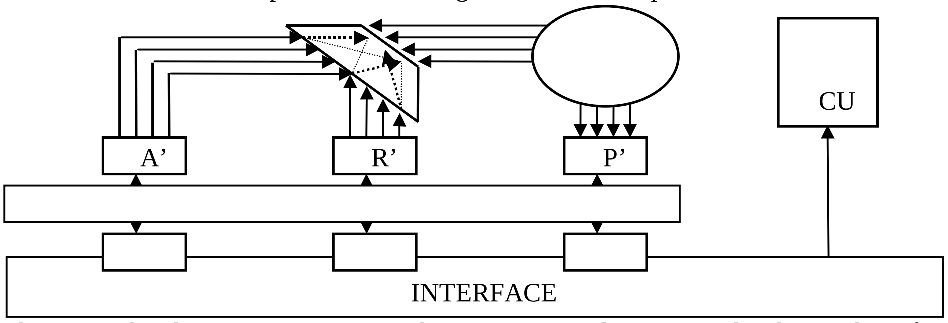


Connections from output layer to input layer can be useful - if there were also connections in input layer between neurons representing A and neurons representing R - one could ask questions about every component of relation (also about A and R)

There are two more data series on chart 3. Results of the network are even better, when as the right answer one treats pattern P with the shortest Hamming distance to the answer of output layer. The results are shown by series OPT. The series LOS shows the modification of this algorithm, in which patterns are not orthogonal but random.

3.5. Knowledge database with Hopfield network

If the information in output layer of the network has some errors, one can use neural network as a filter. The Hopfield network is good to solve this problem:



This network is learning P patterns, at the same time, as layer network. The number of patterns, which one succeeds to teach the network is not big - It is shown in chart 1 (series Hopfield). In this case also network capacity is limited by a number of orthogonal patterns P, which for the settled DL with used encoding is DL/3 (point 3.1).

4. Conclusion

Knowledge database - one of the parts of the expert system - can be implemented in a neural network. The best results were obtained by using in this aim BAM network with single signal sending.

This work is rather a set of ideas, that can be developed. The improvement of network results may be achieved by creating patterns not at random, but on basis of taught relations. One can also improve filtering of noises, which are given by BAM network. Neural networks are good for this aim, but presented Hopfield network has too low capacity. The most interesting results can be obtained by implementing the whole expert system in the similar way.

References:

[1] Hinton G.E., McClelland J. L., Rumerhart D. E.: Distributed Representation, The PDP Perspective

[2] Kosko Bart: Adaptive bidirectional associative memories. Applied Optics, 1987 vol. 26

[3] Osowski Stanisław: Sieci Neuronowe. Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 1994

[4] Schalkoff Robert J.: Artificial Intelligence: An Engineering Approach, Mc Graw-Hill Inc, New York 1990