ISSN 2278-3091
Volume 9, No.4, July – August 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse360942020.pdf
https://doi.org/10.30534/ijatcse/2020/360942020

Representation of Concept Based Censor Production Rules using Neural Networks

Nabil M. Hewahi,
Computer Science Department, University of Bahrain
nhewahi@uob.edu.bh

ABSTRACT

In this paper we present an approach for representing the Concept Based Censor Production Rules (CBCPR) using Neural Networks (NN). CBCPR is a very useful rule structure where it can be used either as standard rule or as Censor Production Rule (CPR) used in real time systems applications. In the proposed approach, NN representation should be able to handle various UNLESS slots related to one CBCPR. The representation should also reflect the PRIORITY slot in the CBCPR to express the priorities of UNLESS slots. A special algorithm to perform the representation and how the knowledge base system should be treated as one NN are presented. For the learning purpose, the backward pass of the backpropagation algorithm can be adopted, whereas the forward pass can be replaced with our proposed approach since it has a direct relation with the representation mechanism.

Key words : Censor Production Rules, knowledge representation, rule Based System

1.INTRODUCTION

Concept Based Censor Production Rules (CBCPR) [6] are rules obtained through a series of developments starting from the standard rule structure in the form of IF <condition> THEN <action> passing through Censored Production Rules (CPR) [11], Hierarchical Censored Production Rules (HCPR) [12], General Structure Rules (GSR) [5] and ending with CBCPR [6]. In rule-based systems, sometimes we have incomplete or inconsistent knowledge, and to overcome this drawback, machine learning is one of the potential solutions. Nowadays Neural Network (NN) and especially deep NN is becoming one very important model in machine learning. In this paper we propose an approach to represent CBCPR using neural networks, this will be very useful in rule -based systems that need to improve their classification. This would be significant when the system is having incomplete or inconsistent knowledge, and also in hybrid systems that incorporate learning in expert systems.

1.1 Pre CBCPR-Structure

CBCPRs [6] are rules basically based on the concept of CPRs [11] which have the form IF <condition> THEN <action> UNLESS <censors>. The condition/antecedent and action/consequence parts work the same way as in standard rule structure, the censors in the UNLESS part are conditions which rarely occur, and that is why such kind of rules can be used in real time systems, if more time is given, more censor conditions can be checked to be more certain about the answer, otherwise, the action will be taken without checking all the censor conditions. If any of the censor conditions is true, the rule action will be prevented/denied. As an example,
(IF Joh-at-home THEN John-watching-TV: 0.8) UNLESS (has-guest:0.04, TV-malfunctioning:0.02)

The certainty value for the above rule without checking any of the censor conditions is 0.8 (this value is usually indicated by γ). If there is a time and the censor condition has-guest can be checked and proved to be false, the certainty value becomes 0.84 (0.8+0.04) (this value is usually indicated by δ). If we still have more time, the second censor condition can be checked and the overall certainty value becomes 0.86 (0.84+0.02). If any checked censor condition is true, the action John-watching-TV is denied. From the previous example we can understand that when John is at home, he usually watches TV unless he has a guest or the TV is not working properly. This means, if we have time to be surer about the rule result, we can check if John has a gust or not. If he has a guest, the rule will fail and the conclusion will not be taken, if he does not have a guest, and we have no more time, we can take a decision with John-watching-TV with 0.84 certainty value. Similarly, we will do the same with TV-malfunctioning censor condition. The more censor conditions are checked (having more time), the rule certainty value will be increased. HCPR is same as CPR, but in addition to the three main slots of the CPR (IF, THEN and UNLESS), two more slots have been added, i.e., GENERALITY and SPECIFICITY, where GENERALITY is to have the name of the parent rule of this current rule and SPECIFICITY is the name of the most specific rule of this current rule. Set or related rules with GENERALITY and SPECIFICITY form a tree called HCPR-tree [12]. This will allow the user to conclude results based on how much specific/general level he/she wants. Below is a set of HCPRs representing rules 0, 1 and 2 and indicated by R0. R1 and R2 are related to HCPR-tree in Figure 1.

R0:
IF [X-eats, X-drinks,X-Reproduce]
THEN [X-Animal]
UNLESS [X-dead]
GENERALITY[ ]
SPECIFICITY[R1, R2]

R1:
IF [X-lives in jungles, X-lives in desert]
THEN [X-is-Wild]
UNLESS [X-in-house]
GENERALITY[R0]
SPECIFICITY[R3, R4,R5]

R2:
IF [X-lives at home]
THEN [X-is-Pet]
UNLESS []
GENERALITY[R0]
SPECIFICITY [R6,R7,R8]

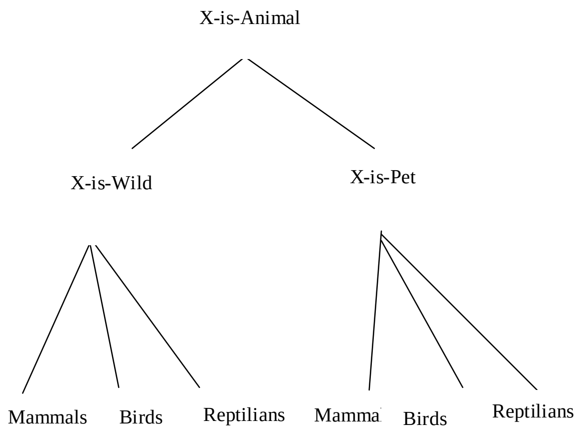

Figure 1: A simple HCPR-trees

In Figure 1, the tree root (level 0 of the tree) represents R0, level 1 in the tree represents rules R1 and R2, and level 2 represents rules R3, R4, R5, R6, R7 and R8 in order from left to right (if any and so on).
GRS [5] introduced a new slot called ALTERNATIVELY to state in the rule the possible next rules to be fired if the current rule is failed. This will help the system to expedite the process of the system inference.

1.2 CBCBR Structure

CBCPR is a rule structure that has been proposed as an extension of CPR, the extension has two directions, the first direction is to introduce a slot to the rule to reflect the main concept of the rule (about what is the rule), the idea of rule concept is that more than one rule might have the same condition/s but are related to different concepts to conclude. The second direction is that in the original CPR, there is only one slot for UNESS, whereas in CBCPR, there might be more than one UNLESS slot, each contains only certain censor conditions related to a specific category. This would be useful to expedite the process of inference by instead of trying to check all the censor conditions (if there is only one UNLESS slot), only censor conditions related to a certain category are checked. This will reduce the required time and allow the system to respond within time limits with higher certainty. To help controlling various UNLESS slots, PRIORITY slot is added to the rule to list the priority of UNLESS slots. The general structure of CBCPR is as below:

```
([Concept-Title]: IF condition
              THEN action
              UNLESS-1 [ Type-1]: [c1,c2,…cn]
              UNLESS-2 [Type-2]: [s1,s2,…,sm]
               .....
               UNLESS-z [Type-z]: [k1,k2,...,kr]
                PRIORITY [p1,p2,..pz]: Ɣ, δ)
```

It is to be noticed that p1, p2 and so on are numbers of UNLESS slots according to their priority in the rule. Let us consider the following CBCPR as example
Rule 1:

```
([ENTERTAINMENT]: IF X is summer
                  THEN X is entertaining
                   UNLESS-1[Personal-Failures]: [X
                   is sick, X has emergency case]
                   UNLESS-2[External-Causes]: [visa
                   not issued, no available flight]
                   PRIORITY[1,2])
```

In Rule 1 example, we have two UNLESS slots for two different categories, Personal-Failures and External-Causes. The PRIORITY slot says that UNLESS-1 censor conditions are having higher priority than UNLESS-2. If in PRIORITY slot, 2 is given before 1, it means UNLESS-2 censor conditions are having higher priority than the censors in UNLESS-1. Ɣ in CBCPR is exactly as in CPR which is the obtained certainty value for the action by checking only the input conditions without looking to any of the censor conditions. δ is the certainty value of the action if the input conditions are true and the tested censor conditions are false based on the given time for the system response. For more information about how these values are calculated, reader can refer to [6]. A more tuning and restructuring for CBCPR is presented in [8].

1.3 Neural Networks for Rule Structure

Due to the importance of neural networks as a machine learning model, many attempts were tried to represent various rule structures using neural networks. Knowledge Based Neural Networks (KBANN) [13] is an algorithm used to make the system learn set of symbolic rules using NN, to perform this task, rules first need to be represented in the form of NN, then the NN is trained using the KBANN followed by the backpropgation algorithm used usually for the feed forward NN. Bharadwaj and Silva (1998) proposed an approach to integrate HCPR with NN called Variable Precision Neural Logic (VPNL). Later Hewahi [7] introduced an approach called General Rule Structure Neural Logic (GRSNL) network to represent GRS in a neural network model. Due to the importance of CBCPR and its comprehension compared with CPR, in this paper we present an approach to represent it using NN.

2.THE PROPOSED APPROACH TO REPRESENT CPCBR USING NN

CBCPR structure is not simple because it has various slots related to UNLESS, also there should be a technique to make the representation of PRIORITY slot viable. The other important issue is that how to deal with Ɣ and δ. To make the process clear, we shall consider the following template

```
Concept  IF   condition
            THEN action
               UNLESS-1 [ …]
                ..
               UNLESS-n […..]
               PRIORITY […..]
```

Logically the action should be in output layer, whereas, the input conditions and the input censor conditions should be in the input layer. All censor conditions related to one UNLESS slot should be connected to one unit in the hidden layer, this means if we have n UNLESS slots related to one rule, it means those n neurons in the hidden layer should be connected somehow to one neuron (to represent all the UNLESS slots related to one CBCPR in one node) then from this neuron a connection to the rule output node should be there. To visualize this process, let us consider Figure 2. As shown in Figure 2, I1, I2 and I3 are the rule input conditions, C11, C12 and C13 are the censor conditions for UNLESS-1slot, C21, C22 and C23 are the censor conditions for UNLEASS-2 slot and so on. In the hidden layer 1 all the inputs I1, I2 and I3 are linked to one neuron (oval shape), and all the censor conditions of UNLESS-1 slot are linked to one neuron (square shape). Similarly, all the censor conditions of UNLESS-2 slot are linked to one neuron and so on. In hidden layer 2, all the UNLESS slots in layer 1 are linked to one neuron. In the output layer, the neuron from hidden layer 1 related to condition inputs and the neuron of the hidden layer 2 represents all the UNLESS slots are linked to the output neuron. To consider the representation of PRIORITY slot, p1, p2 and p3 in Figure 2 just to represent the priorities of UNLESS slot nodes in the rule. It is to be noticed that neuron of p1 priority is having two outputs, the first (RO1) goes as input to neuron of UNLESS slot with priority p2 and the other output (RO2) goes to the triangle neuron which deals with all the UNLESS slots. Similarly, is the relation between the node with p2 priority and the node with p3 priority and so on if there is more UNLESS slots. Actually, for simplicity we consider rectangle shape of UNLESS slot of p2 in hidden layer 1 despite it should be considered in layer 2 because the output of p1 goes as input to p2. Similarly, for other UNLESS slots based on their priority. The working process of these two outputs of the UNLESS slot will be explained in the algorithm.

Our assumption is that the possible input values for input conditions are 1, -1 and 0 for true, false and unknown respectively. Also, for the sensor conditions, the possible input values are 1, -1 and 0 for true, false and unknown respectively. Since the censor conditions are rarely occur, their initial weights should be very small, two choices can be followed, the first we use the value of the certainty of the censor condition as the weight for its link with the neuron in the first hidden layer, the second choice is to multiply the certainty value with a factor f smaller than 1, this will make the maximum initial weight of the censor condition as its certainty value. To make it clear, assume UNLESS-1[c1:0.06, c2:0.03], it means the weights of the links will be

First approach

w1=-0.06   w2=-0.03

Second approach assuming f = 0.2

w1 = 0.06 * 0.2 = 0.012   w2= 0.03 * 0.2*1 = 0.006

A third approach can also be used by changing f based on the importance effect of the censor condition. Our assumption is that, all the censor conditions are listed based on the certainty value of the censor conditions from higher to lower as given in the UNLESS-1 in the above example. Starting with f=0.2 and using the same mechanism in the second approach

w1 = 0.06 * 0.2 = 0.012

new_f = old_f /co,   (1)

where co is the censor order

new f = 0.2/2 = 0.1

w2 = 0.03 * 0.1 = 0.003

The process will continue up to the end of all the censor conditions in the UNLESS slot.

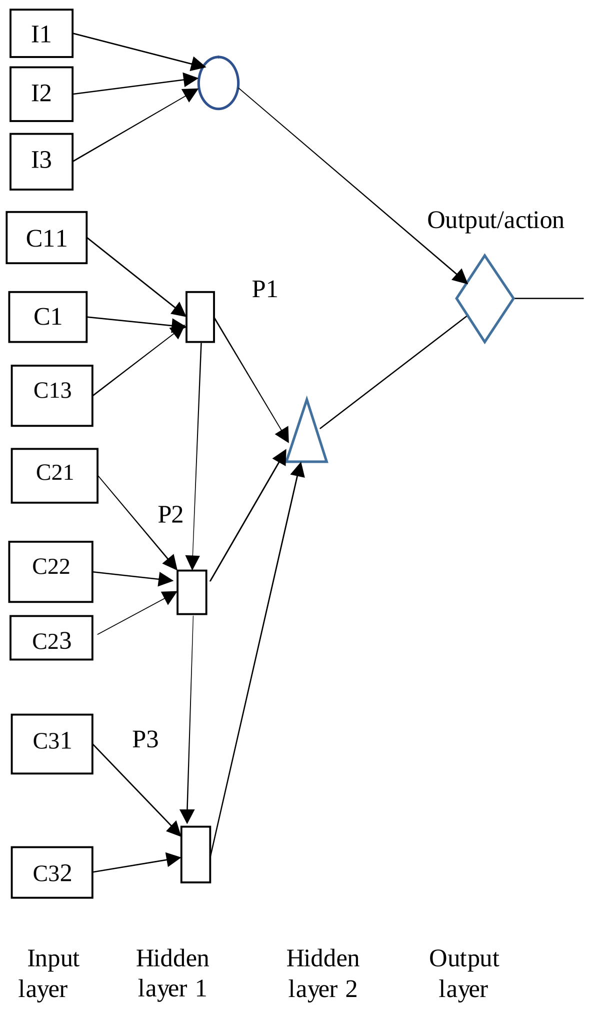

Figure 2: Simple representation of one CBCPR using NN

The algorithm used to construct the neural networks for CBCPR is as below:

- For all the rules in the system, include all the input conditions and the censor conditions as inputs in the NN input layer.
- Consider 1, -1 and 0 as inputs for condition inputs for true, false and unknown respectively. Also, consider 1, -1 and 0 as inputs for censor conditions for true, false and unknown respectively. The output of the neural network is either 0 (false) or a value greater than 0 representing the certainty value of the rule.
- Connect all the condition inputs to one neuron in the first hidden layer (oval shape), the initial weights are 1's.
- Connect all the censor conditions in one UNLESS slot to one neuron in the hidden layer 1 (rectangular shape). The weights are the certainty values of the censor conditions as explained above.
- Connect all the neurons of rectangular shapes (related to UNLESS) slots to one neuron in the second hidden layer (triangle shape). The initial weights for the connections are 1's.
- The rectangle node related to UNLESS slot with priority p1 has two outputs, one goes to triangle node and the other goes as input to the rectangle node with priority p2. Similarly, the node with p2 has two outputs and so on. The last UNLESS slot rectangle node will have only one output which goes as input to the triangle node.
- Connect the neuron of the first hidden layer related to the input conditions and the triangle shape node in the second layer with the output node in the output layer related to the rule action. The initial weights for the connections are 1's.

Now we need to explain how is the output of each neuron is computed

a. Hidden layer 1 (oval shape node): The computation of this node is similar to normal computation used in neural networks but with a specific condition

If any input is -1 (false), then the node output (OO) is -1

Otherwise, OO = $\sum W_1 . I$ (2),

where W_1 is the vector of weights from inputs conditions to the node (initial weights are 1's), and I is the vector of condition inputs.

b. Hidden layer 1 (rectangular shape node p1 as example):

If any input is 1 (true), then the node output which goes to p2 node (RO1) is -1, and the second output which goes to the triangle node (RO2) is -1.

Otherwise, RO1 = 1 and RO2 = $\sum W_2 . (-C)$ (3)

where C is the vector of censor conditions and W_2 is the vector of weights from the censor conditions to the node. We multiply – with C to make the value equal to 1 because in this case all the values in C are either -1 or 0 (the value is false which means -1 and that is why we multiply it with – to get correct certainty). It is also to be noted that the weights of the links that connect p1 to p2 and p1 to the triangle node are 1's.

c. Hidden layer 1 (rectangle shape node p2 as example):

If any input is 1(true) except the value of RO1 coming from p1(if any input is 1, we do not care about the value of RO1 coming from p1), then the node output goes to p3 with RO1=-1 and RO2=-1

Otherwise (all inputs are -1 and RO1=1 coming from p1), RO1 = 1 and RO2 = $\sum W_2 . (-C)$, it is to be noted that RO1 of p1 is not included in the computation, it is just used to decide the value of RO2 of the next node.

d. Hidden layer 2 (triangle shape node):

If any of the inputs to this node is -1, then the output (TO) is -1,

Otherwise, TO = $\sum W_3 . VRO2$ (4)

Where W_3 is the vector of weights from layer 2 to layer 3 (initially 1s). VRO2 is a vector containing all RO2 values obtained from the rectangle node related to UNLESS slot.

e. Output layer (diamond shape node):

If any of the inputs (coming from the oval node and from the triangle node) is -1, then the output (OUT) is 0 (false),

Otherwise, OUT = $\sum (W_4 . TO)/n$ (5)

Where W_4 is the vector of weights from the second hidden layer (initially 1s), and n is the number of UNLESS slots in CBCPR. Equation (5) is based on the computation of certainty factor for CBCPR in [6]. The result of OUT will be the certainty value of the rule. Figure 3 shows a flowchart that summarizes the general procedure to compute the output of the neural networks.

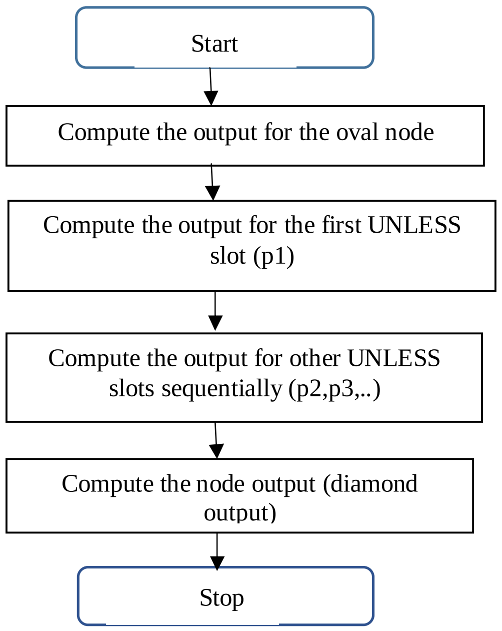
Start
Compute the output for the oval node
Compute the output for the first UNLESS slot (p1)
Compute the output for other UNLESS slots sequentially (p2,p3,..)
Compute the node output (diamond output)
Stop

Figure 3: Flowchart for the sequence of computation in the NN

2.1 Knowledge Base Representation

In rule-based systems, the knowledge base contains set of rules, in our case we shall consider set of CBCPRs. Two basic cases can be considered:

a. Independent CBCPRs: In this case each CBCPR might have independent output and each rule does not depend on any other rule. Figure 4 illustrates this situation. NNR_i in Figure 3 and in any later figure means the NN for the ith CBCPR. In Figure 3, NNR_1 and NNR_2 have two independent outputs. Independent means the firing of one rule does not depend on firing of another rule.

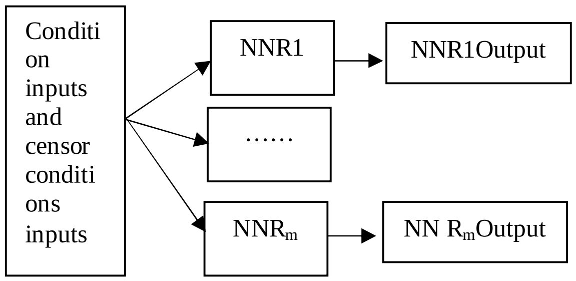

Figure 4: Knowledge base with independent CBCPRs

b. Dependent CBCPRs: In this case some CBCPRs depend on previous fired CBCPRS. Figure 5 shows this situation. In Figure 5 CBCPR number n depends on CBCPRs 1 and 2.

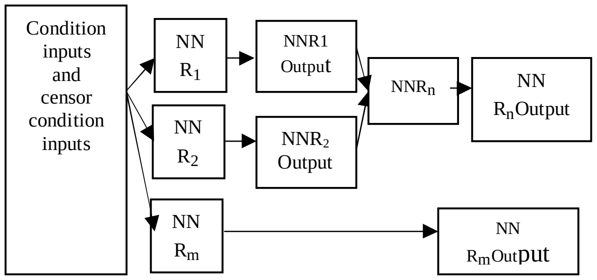

Figure 5: Knowledge base with dependent CBCPRs

2.2 Learning Process Discussion

Once the knowledge base is represented, the main question is what would be the proper approach to perform the learning process. Rule based machine learning system is a system that utilizes machine learning techniques to learn rules. Some of such kind of systems are artificial immune system which is based on modeling the biological immune system [4] [12], classifier system which is based on genetic algorithms [9][10], and associations rules which are based on discovering the relations between variables in the dataset [1]. However, the previous stated approaches do not depend on NN representation. As stated before, one of the approaches used to train rules using NN structure is KBANN algorithm, the other choice is to use backpropgation algorithm. Whatever is the approach to be followed, the system needs a dataset of examples to be applied on the knowledge based NN structure. In the dataset, enough number of examples with various cases for input values related to condition inputs and censor condition inputs is required to train the system. Table 1 shows a simple example of a small segment of a dataset. In Table 1, i refers to condition input, c refers to censor condition input, and O represents the corresponding output. AT in the table is the "on average" number of censor conditions permitted to be checked on some given time. We mean by "on average" is the average number of censor conditions in the UNLESS slots in all the participated CBCPRs in the inference mechanism. For example, if AT is 2, it means only maximum of two censor condition inputs in any of UNLESS slots of any participated CBCPR in the inference are checked. This might not be very accurate, but it is a sort of heuristic.

Table 1: An example of a segment of a dataset that might be used for training

EX	i1	i2	i3	i5	c1	c2	c3	c4	AT	O1	O2
1	1	1	-1	1	0	0	-1	0	1	0	0
2	1	1	1	0	1	1	-1	0	2	0.5	0.6
3	0	0	0	1	1	1	0	1	1	0.8	0
4	-1	1	1	0	0	0	0	1	3	0	0.83

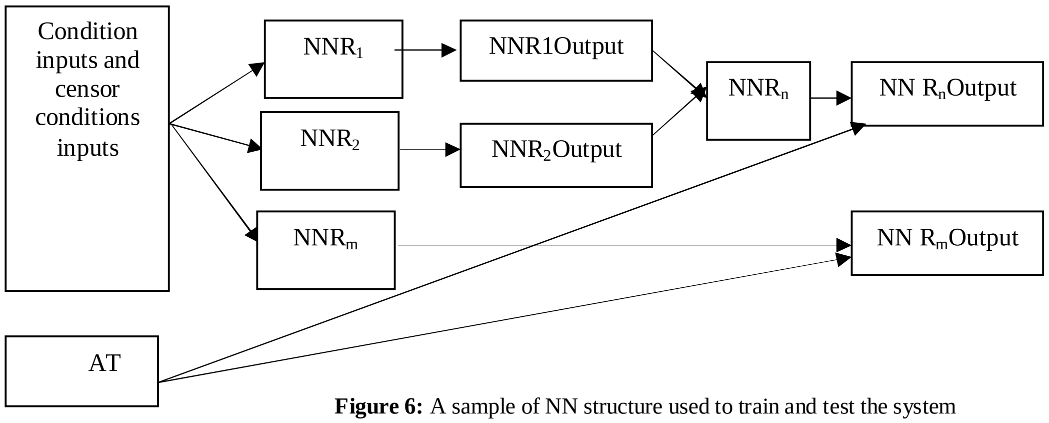

Figure 6: A sample of NN structure used to train and test the system

It is to be noted that the condition inputs and the censor condition inputs are related to the whole knowledge base. According to what has been given in the algorithm, the result is either a zero or a value greater than zero that refers to the certainty value. Based on the concept presented in Table 1, a sample example of NN used for training and testing can have a structure shown in Figure 6.

Based on the above mechanism, backpropgation algorithm can be easily used to train the knowledge-based system, the only difference is that the forward pass of the backpropgation algorithm must be replaced with our given approach in section 2 for assigning weights and getting the output. The initial weights of AT to output nodes are generated at random. The second pass of the backpropgation algorithm to adjust the weights can still be a valid procedure.

3.CONCLUSION

CBCPR is considered to be a well-structured rule, and can work well in normal cases or in real time systems. Adaptive rule-based systems are becoming very important in various applications. In this work, an approach to represent CBCPR in the form of NN is presented. Also, a NN representation of a knowledge based on CBCPRs system has been proposed. A mechanism to learn and produce outputs based on training inputs has been explained. The representation of CBCPR is not simple or straight forward process because the representation has to handle various UNLESS slots in CBCPR. The more is the UNLESS slots, the more is the complexity of the NN. This complexity emerges to maintain the PRIORITY slot in the CBCPR. This research opens the door to have very effective adaptive rule-based system based on NN that can be used in various real time applications such as robotics navigation and industry, or in normal applications such as expert systems. Some of the future directions would be applying the CBCPR representation and the proposed approach for computing the outputs of the rules to real applications in various domains, this should be done for real time systems and non-real time systems.

REFERENCES

1. R. Agrawal, T. Imieliński. A. Swami. **Mining association rules between sets of items in large databases**, *Proceedings of the 1993 ACM SIGMOD international conference on Management of data - SIGMOD '93*, 1993, pp. 207-216. https://doi.org/10.1145/170035.170072
2. K.Bharadwaj and N.Jain. **Hierarchical censored production rules(HCPR) system**, *Data and Knowledge Engineering*, Vol.8, pp.19-34, 1992.
3. K.Bharadwaj, and J.Silva. **Towards Integrating Hierarchical Censored Production Rule(HCPR) Based System and Neural Networks**, in Lecture notes in *Artificial Intelligence, 1515*, Oliveira, 1998, pp.121-130.
4. de Castro and N. Leandro, Timmis, and Jonathan. **Artificial Immune Systems: A New Computational Intelligence Approach.** Springer, 2002. pp. 57–58. 5. N. Hewahi, **A general rule structure**, *Journal of Information and Software Technology*,44, pp.451-457, 2002.
6. N. Hewahi. **Concept Based Censor Production Rules**, *International Journal of Decision Support System Technology*, Vol. 10, issue 1, pp. 59-67,2018.
7. N. Hewahi, **Principles on Integrating General Rule Structure(GRS) Based Systems and Neural Networks**, *Proceedings of the 2004 International Conference on Artificial Intelligence (IC-AI'04), Las Vegas, Nevada, USA*, June 21-24,2004, pp. 1-12.
8. N. Hewahi,. **Restructuring Concept Based Censor Production Rules**, *2018 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT), Bahrain*, 2018, pp.1-3. https://doi.org/10.4018/IJDSST.2018010104
9. J. Holland. **A Mathematical Framework for Studying Learning in Classifier Systems**, *Physica D*,22,pp. 307-317, 1986.
10. J.Holland. **Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel**, in *Machine Learning: An Artificial Intelligence Approach Mitchell, Michalski and Carbonell, Eds.*, Vol. 2, Chapter 20, Los Altos, CA Morgan Kaufmann, pp. 593-623, 1986.
11. R. Michalski and P. Winston. **Variable Precision Logic**, *Artificial Intelligence*, 29, pp. 121-145, 1986. https://doi.org/10.1016/0004-3702(86)90016-0

.12. M. Read, P. Anders, and J. Timmis. **An Introduction to Artificil Immune Systems**, in Handbook of Natural Computing , Rozenberg, Back, Kok. Eds. Berlin, Heidelberg, Springer, 2012.

.13. G. Towell and J. Shavlik. **Knowledge Based Artificial Neural Networks**. *Artificial Intelligence*, 70, No. 1-2, pp. 119-165, 1994. https://doi.org/10.1016/0004-3702(94)90105-8