International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:14, No:8, 2020

Rule Insertion Technique for Dynamic Cell Structure Neural Network

Osama Elsarrar, Marjorie Darrah, Richard Devin

***Abstract*—**This paper discusses the idea of capturing an expert's knowledge in the form of human understandable rules and then inserting these rules into a dynamic cell structure (DCS) neural network. The DCS is a form of self-organizing map that can be used for many purposes, including classification and prediction. This particular neural network is considered to be a topology preserving network that starts with no pre-structure, but assumes a structure once trained. The DCS has been used in mission and safety-critical applications, including adaptive flight control and health-monitoring in aerial vehicles. The approach is to insert expert knowledge into the DCS before training. Rules are translated into a pre-structure and then training data are presented. This idea has been demonstrated using the well-known Iris data set and it has been shown that inserting the pre-structure results in better accuracy with the same training.

***Keywords*—**Neural network, rule extraction, rule insertion, self-organizing map.

I. Introduction

ARTIFICIAL Intelligence plays a key role in developing devices that can analyze situations like a human. Developing systems with a set of guiding knowledge that is then able to learn from new experiences to refine that knowledge is key to simulating human decision making. Neural-Symbolic learning systems play a key role by combining the benefits of both the neural and symbolic paradigms of artificial intelligence [1].

Accuracy and confidence is very important for safety-critical uses of neural networks. The rationale for using rule insertion is that expert knowledge represented in a set of rules, which could possibly be incomplete or incorrect due to insufficient knowledge, can be inserted to initialize a neural network before training is applied. The initial knowledge is inserted using a rules-to-network algorithm. The initial symbolic knowledge that is inserted becomes the initial neural network structure. This process creates a "neural-symbolic" system utilizing a combination of theoretical and empirical data. The initial symbolic knowledge then goes through a stage of training and refinement. Upon completion of training, rules are extracted again for comparison. This three-step process assists in ensuring the most accurate output, reduces training time, and provides confidence by allowing developers and users to better understand the internal workings of the neural network through inspection of the rules.

Neural networks are not recognized for their capacity to use symbolic knowledge, but rather from their capability "to be trained from data". They have become an acknowledged tool in machine learning toolboxes. Usually, neural networks "readily" store knowledge in distributed internal weights, not in symbolic form. Although neural networks are commonly used for generalizations, other applications may require the knowledge be used in symbolic form [2]. Therefore, investigation into the interchange of information between connections and symbolic representations is necessary for effective learning.

Kurd et al. [3] discussed that the dilemma with the use of artificial neural networks in a safety critical situation is that the software lifecycle relies on determining the specifications at the initial phase of development. This is not supported if the neural network starts with no initial internal structure, which is the case with the DCS self-organizing map that is the focus of our research. The lifecycle of the hybrid systems like the one we are suggesting can be described by the "W" model (Fig. 1) [3].

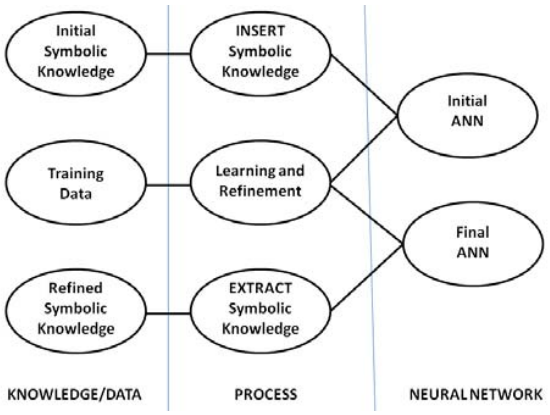

Fig. 1 Rule Insertion/Extraction "W" Model [3]

In Fig. 1 the following levels are depicted:

- **Symbolic Level:** This level is associated with symbolic information and deals with analysis in terms of symbolic knowledge.
- **Translation Level:** This level is where symbolic knowledge and neural architectures are joined or separated.
- **Neural Learning Level:** This level uses neural learning to adapt and refine symbolic knowledge.

In the past, others have explored the idea of combining rule-based knowledge and neural learning. Towell and Shavlik [4], [5] introduced the new algorithm named Knowledge-Based Neural Network (KBANN). They felt that this algorithm

Marjorie Darrah is with the West Virginia University, United States (e-mail: marjorie.darrah@mail.wvu.edu).

would improve the learning speed because it is not ignoring any information. They described this algorithm as a way to address the problems of training "deep" networks. Fig. 2 shows the process that Towell and Shavlik used for Rule Insertion. KBANN is a hybrid learning system and is more effective at classifying examples compared to other machine learning algorithms. Unfortunately, the networks created by KBANN, known as KBANN-nets, have "deep" network properties that are not well suited to work with backpropagation. To address this issue, the Desired Antecedent Identification (DAID) algorithm was introduced.

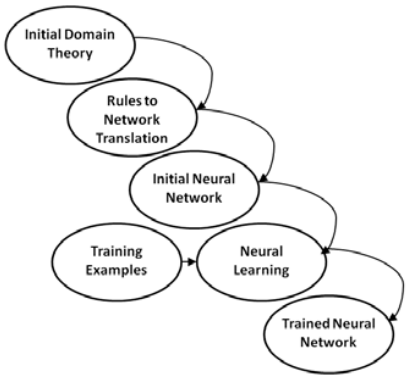

Fig. 2 Towell and Shavlik [4] method for Rule Insertion

The DAID was motivated by two observations. First, the "deep" neural networks cause trouble to the neural leaning techniques because error signals become diffused. Second, it had been shown that KBANN is most effective when antecedents are ignored by the network. The DAID aids in this issue by lessening error. Ultimately the DAID is most useful in deep structures due to its learning bias towards learning at the bottom, whereas backpropagation is most useful in shallow structures due to its bias towards learning at the top of chains.

Another idea from Giles and Omlin [6] discusses methods for extracting, inserting and refining symbolic grammatical rules for recurrent networks. The issues also discussed in this paper include how rules are inserted into the recurrent neural network, how training and generalization is affected, and how the rules can be checked in order for correction. The method Giles and Omlin [6] devised requires the network size to exceed the number of Deterministic Finite State Automata (DFA) states. It was expected that the training time would decline with rising rule strength, but the network does not easily recognize partial correct rule insertion if the rule strength is too great.

An additional aspect of symbolic knowledge extraction and insertion is rule checking, allowing for the establishment of the validity of the knowledge. Rule checking compares rules extracted from trained networks with prior knowledge. However, rule checking becomes increasingly difficult with rising rule strength when incorrect rules are inserted into a network. Further, Giles and Omlin [6] suggest that network architecture can be altered during training with symbolic guidance, and symbolic information gained from under-trained networks could prove useful in determining the current network architecture.

This paper presents an approach for inserting rules to a specific neural network structure, the DCS neural network that has been used in several safety critical applications including adaptive aircraft control [7] and on-board health state awareness for Unmanned Aerial Vehicles (UAVs) [8]. Section II discusses the process by outlining the DCS structure, the rule extraction process and the rule insertion process. Section III discusses the application of the process to a common benchmark data set. Section IV provides the result of the experiment and Section V provides some conclusions that can be drawn.

II. The Process

A. Structure of the DCS Neural Network

As previously mentioned, one type of self-organizing map is called the DCS neural network [9]-[11]. The DCS is designed to learn and represent the topology of the input space. After training on data, the DCS structure is arranged like a Voronoi diagram (Fig. 3) [12].

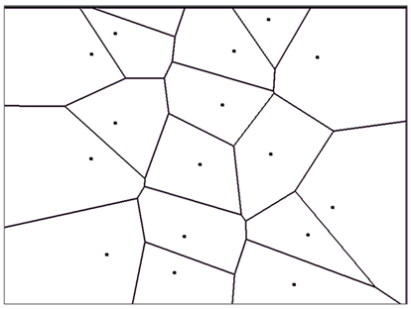

Fig. 3 Voronoi Diagram

The centroids of the Voronoi regions in Fig. 3 represent the reference vector for each of the cells. These centroids are the neurons in the neural network. The neurons, c_{ij}, of neighboring cells are then connected using a Delaunay triangulation [12].

Given an input to the DCS, v, the best matching unit (BMU) is the neuron whose weight, w, is closest to v, and the second best matching unit (SEC) is the neuron whose weight is the second closest to v. Along with the BMU, the neighbors of the BMU are found through the Delaunay triangulation, which connects the centers of the Voronoi regions if they share a boundary. During training and the presentation of data, adjustments are made to the BMU and SEC neurons [13].

The DCS algorithm consists of two types of learning rules, Hebbian (1) and Kohonen (2) and (3). The learning rules described in the equations allow the structure to change based on the inputs it is presented. Neurons are added and neuron positions are adjusted based on the data. This allows the evolution of the network and the potential for many arrangements [12].

$$c_{ab} = \begin{Bmatrix} 1\ a \in [BMU, SEC]\ \wedge b \in [BMU, SEC] \\ \alpha \cdot c_{ab}\ \alpha \cdot c_{ab} > 0 \\ 0\ \alpha \cdot c_{ab} < 0 \\ 0\ a = b \end{Bmatrix} \quad (1)$$

$$\Delta w_{BMU_i} = \varepsilon_{BMU}(v_i - w_{BMU_i}) \quad (2)$$

$$\Delta w_i = \varepsilon_{NBR}(v_i - w_i) \quad (3)$$

Below, the rule extraction process, DCS structure to Human Understandable Rules, is discussed first, since this process was established for the DCS neural network in previous work [13]. Then we discuss how the rule insertion process, Human Understandable Rules to DCS structure, would work.

III. Rule Extraction

A negative seen when using neural networks is the fact that the knowledge acquired during training is coded as weights or activation values. This results in very few tools capable of validating neural network techniques. By using rule extraction, a developer can, at least in part, determine the internal knowledge of the trained neural network and validate that what has been learned matches expert understanding and intended need [14].

Rule Extraction techniques have been developed for many neural network types [15]-[17]. This is a process that can help make neural network output more understandable by representing the internal knowledge of the neural network as a set of rules. The predictions or classifications of the network can be explained through the rules extracted from it, making neural networking less of a black box of unexplained answers and more of an understandable process [18]. Accuracy of the rules is generally judged by their agreement with the neural network [19].

The process of extracting a list of human readable rules from the cell list output of the DCS neural network is straight-forward. Each data point is assigned a BMU; the BMU is a centroid of a Voronoi region (cell). Then for each cell there is a list of points that are assigned to that region. From this list of points the minimum and maximum values are determined in each dimension and these values are used to create a bounding box in the parameter space. This bounding box is the smallest such n-dimensional box that contains each point in the cell. Each rule is simply a list of the boundaries of these bounding boxes. In pseudocode the algorithm is as follows:

```
For each(cell in cells):
        For each(datapoint in cell):
            For each(param in datapoint):
                maxes[cell,param]=max(maxes[cell,param],
                datapoint[param])
                mins[cell,param]=min(mins[cell,param],
                datapoint[param])
```

The following is an example of a list of extracted rules. The data set used to train the DCS in this case was the IRIS benchmark data that will be described later in the paper, with four input variables and three output types.

```
RULES FOR CELL1
IF (sepal_length>=6.7 AND <=7.4) AND
IF (sepal_width>=2.8 AND <=3.6) AND
IF (petal_length>=5.7 AND <=6.1) AND
IF (petal_width>=1.6 AND <=2.5)
        THEN...2

RULES FOR CELL2
IF (sepal_length>=4.3 AND <=5) AND
IF (sepal_width>=2.3 AND <=3.6) AND
IF (petal_length>=1 AND <=1.6) AND
IF (petal_width>=0.1 AND <=0.3)
        THEN...0

RULES FOR CELL3
IF (sepal_length>=6.3 AND <=6.9) AND
IF (sepal_width>=2.5 AND <=3.4) AND
IF (petal_length>=5.1 AND <=6) AND
IF (petal_width>=1.8 AND <=2.5)
        THEN...2

RULES FOR CELL4
IF (sepal_length>=5 AND <=6) AND
IF (sepal_width>=2 AND <=2.9) AND
IF (petal_length>=3 AND <=4) AND
IF (petal_width>=1 AND <=1.4)
        THEN...1

RULES FOR CELL5
IF (sepal_length>=5.5 AND <=6.1) AND
IF (sepal_width>=2.6 AND <=3) AND
IF (petal_length>=4 AND <=4.5) AND
IF (petal_width>=1 AND <=1.5)
        THEN...1

RULES FOR CELL6
IF (sepal_length>=7.3 AND <=7.7) AND
IF (sepal_width>=2.6 AND <=3.8) AND
IF (petal_length>=6.3 AND <=6.9) AND
IF (petal_width>=1.8 AND <=2.3)
THEN...2
```

As mentioned previously, the rules make up the bounding boxes that loosely approximate the n-dimensional Voronoi regions. To illustrate the idea, Fig. 4 is an example of a two-dimensional Voronoi diagram that uses two of the variables, sepal length and sepal width, from the rules in the preceding list. The coordinates of the centroids for these two variables were used to create the Voronoi diagram.

Fig. 4 shows some of the bounding boxes for the extracted rules in the previous list overlaid on the Voronoi regions. We observe how the bounding boxes approximate the cells of the Voronoi diagram, even though it is limited to just two dimensions. It can also be noted at this time that the approximation is not exact, there was a more exact rule extraction method developed [20], however the rules that were the output of that method are not considered human understandable, but were more mathematical.

Such drastic overlapping does not occur when the rules are represented in all four dimensions.

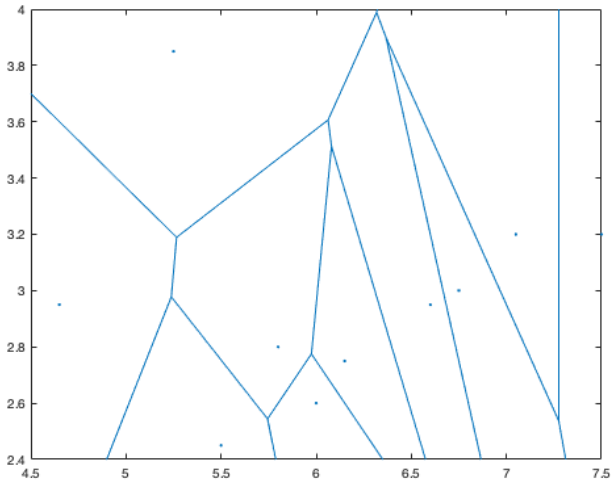

Fig. 4 Voronoi diagram using a 2-dimensional projection of the centroids of the DCS

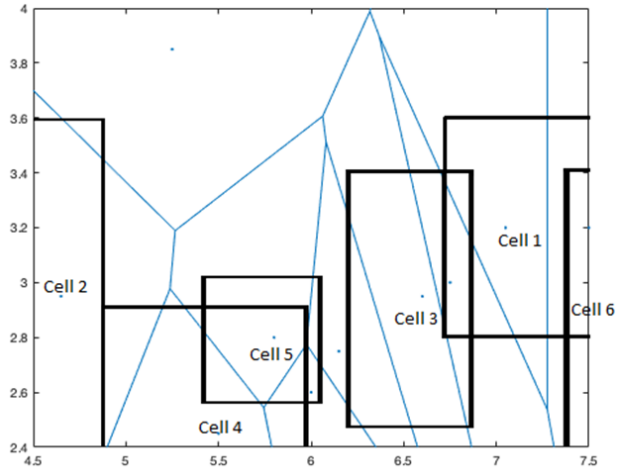

Fig. 5 Voronoi diagram of a 2-dimensional projection of the centroids of the DCS with some of the extracted rules bounding boxes overlaid

IV. Rule Insertion

Rule insertion is the process of supplying internal knowledge to influence the formation of the neural network before training occurs. The knowledge influences the formation towards a potential classification structure, which is used in initializing the neural network, and then trained upon, allowing the rules to be refined.

The hypothesis is that the neural network with rules inserted should be able to be trained faster and be more accurate than the original neural network. The human readable rules can be represented simply as a collection of labeled convex subspaces inside a parameter space, where the label is the category assigned to each subspace. These subspaces are described by a series of if-then statements for each input variable. For example, "if a is less than x and x is less than b AND c is less than y and y is less than d, then the dependent variable belongs to category 1". Using the boundaries of these convex subspaces (in this case a rectangle or bounding box), the rules are converted into a collection of centers for the bounding box or centroids for a Voronoi region. These Voronoi centroids become the neurons of the DCS and provide the initial starting point for neural network training. The DCS usually starts with two or more randomly placed neurons and then either modifies their positions or "grows" by adding additional neurons based on the data.

Now we suppose that the rule list given previously is not the result of training the DCS, but for example was given to us by an expert botanist. Next, suppose we want to insert these rules to give the DCS some prior knowledge on which to train. In this case, we would take each rule and determine the middle values for each parameter. This n-dimensional point then becomes the centroid for a Voronoi region or a neuron of the DCS. The list of centroids is taken and directly used as the initial set of neurons for the DCS. The corresponding centroid list for the previous list of rules would look like:

{7.05, 3.2, 5.9, 2.05}, {4.65, 2.95, 1.3, 0.2}, {6, 2.6, 5.2, 1.7}, {6.75, 3, 4.7, 1.5}, {6.6, 2.95, 5.55, 2.15}, {5.5, 2.45, 3.5, 1.2}

The output 0, 1, or 2 for the rules would also be stored associated with the centroid (neuron).

In order to visualize the data, we use two dimensions and take sepal length and sepal width as the horizontal and vertical axes (respectively). In Fig. 6, we can see the boxes that depict the rules and the centers of the boxes that become the centroids of the Voronoi regions.

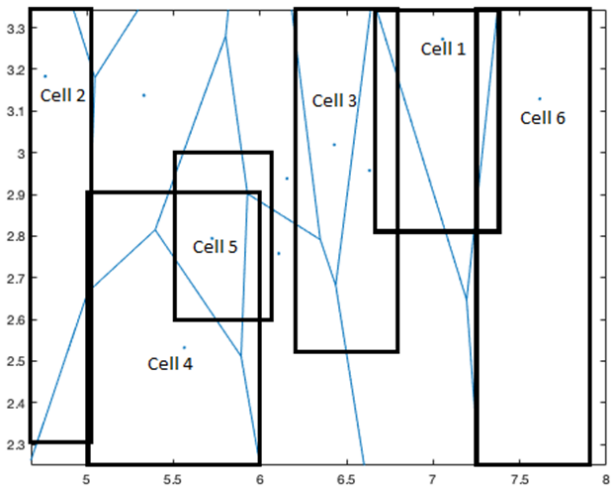

Fig. 6 Rule Set Depicted as Boxes in 2-dimension Projection Overlaid on Voronoi Diagram Constructed from the Box Centers

The centroids given above produce the Voronoi regions in Fig. 7. We note that Fig. 7 Voronoi diagram is not exactly like Fig. 4 Voronoi diagram, but they have some similarity in structure. We recall the structure in Fig. 4 resulted from training and the structure in Fig. 7 resulted from using a set of rules to develop the structure.

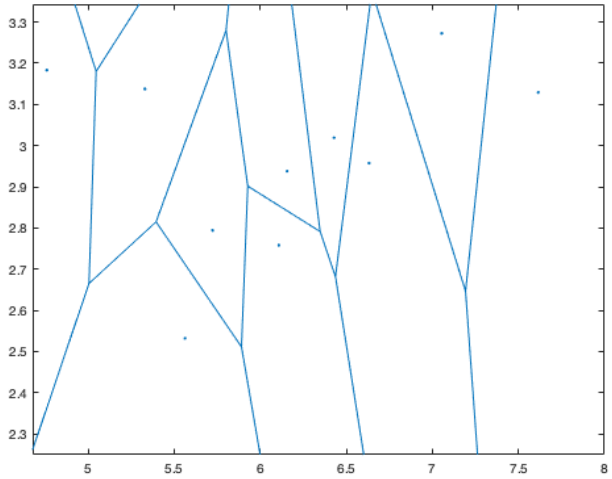

Fig. 7 Voronoi Diagram of a 2-dimension Projection of the Centroids for an Inserted Rule Set

V. Application of Process to Benchmark Data Set

A. IRIS Data Set

One of the most popular machine learning benchmark data sets is the Iris data set. The problem to be solved is to learn which category an Iris flower belongs to based on four measurements: sepal length, sepal width, petal length and petal width. The three Iris categories are: Setosa, Versicolour, and Virginica. The University of California Irvine (UCI) Machine Learning Repository offers many free data sets for testing algorithms. The Iris data set is one of the most popular sets for testing machine learning algorithms. It is composed of 150 instances divided evenly between the three categories (i.e. 50 instances per category.)

B. Comparing the Results

In this section, we will test the efficacy of the rule insertion by first training the DCS neural network with no pre-knowledge (starting with the configuration of two random neurons) and training the DCS with inserted pre-knowledge (rule set inserted into neural network structure as a set of starting neurons). Rule Insertion relies on the processes of engaging an expert to help formulate an initial set of rules. In the case of this proof of concept study using a benchmark Iris data set, no expert was available to construct a set of rules, so a "typical" set of rules was used. The rule set used as the pre-knowledge in the test was similar to rules sets that were extracted; the bounding values for the parameters were approximated in order to provide a starting set of neurons.

For each training epoch, the DCS neural network was trained using a random 75% of the data points from the IRIS data. The remaining 25% of the data points were used to test the accuracy of the resultant neural network. The "neural network accuracy" was calculated as the percentage of data points that were correctly classified by the neural network. In addition, each time the DSC was trained, Human Understandable rules were extracted using the process described earlier. The extracted rules were then tested and the "rule accuracy" was calculated as the percentage of data points that were correctly classified by the set of extracted rules. To avoid overfitting, the DCS was limited to only grow to the size of four neurons, which leads to only four cells.

For testing whether the network would be more accurate being initialized in the default way or initialized with the inserted rules, two experiments were conducted. The DCS NN was developed in both ways, trained using the Iris data ten times, rules were extracted at the end of each training. To compare the two methods of initialization, the accuracy of the trained DCS NN to predict Iris type and the accuracy of the extracted rules to predict Iris type were compiled.

First, the DCS NN was created with the default initialization of two random neurons. The DCS was trained on the Iris data ten different times. When the DCS was trained with the default initialization, the accuracy for the neural network prediction was on average \$92.4 ± 1.86% and the average prediction from the set of the extracted rules themselves averaged \$90.2 ± 1.66%.

Second, the DCS NN was created with the rules inserted. This initialization started with several nodes that were based on the rule set used (same rule set used each time). The DCS was again trained on the Iris data ten different times. When the DCS was trained with the rule-based initialization, the accuracy for the neural network prediction was on average \$94.7 ± 1.57% and the prediction from the set of the extracted rules themselves averaged \$94.2 ± 1.28%. This was an improvement of 2.5% for the network prediction and 2.2% for the extracted rule prediction.

VI. Conclusions

Several methods of rule extraction from the DCS neural network had already been developed [20], but there was no previous rule insertion process investigated. Our research focused on developing a method for inserting rules into a DCS neural network structure. In this paper we determine a method for rule insertion for this type of neural network and tested its usefulness to produce results on a benchmark data set. These findings show that there is great potential for this technique to improve the accuracy of the neural network and also improve the accuracy of any rules extracted. This opens up numerous possibilities for creating more efficient and more accurate neural networks. The initialization of the DCS with "expert" rules allows the neural network to come to a better solution in the same time, than can be developed by just training alone.

This DCS neural network has been used in several mission and safety critical applications, namely adaptive aircraft control [7] and on-board health state awareness for Unmanned Aerial Vehicles (UAVs) [8]. The ability to allow a developer to work with an expert to develop a better Neural-Symbolic system is important to further the usefulness of this neural network type.

References

[1] A. S. Garcez, D. M. Gabbay, K. B. Broda, Neural-Symbolic Learning Systems: Foundations and Applications, Springer-Verleg, 2002.
[2] M. Hoehfeld and S. E. Fahlman, “Learning with limited numerical

precision using the cascade-correlation algorithm," IEEE Transactions on Neural Networks, vol. 3, no. 4, pp. 602-611, 1992.
[3] Z. Kurd, T. Kelly, and J. Austin, "Safety criteria and safety lifecycle for artificial neural networks," In Proceedings of Eunite, vol. 2003, 2003.
[4] G. G Towell and J. W. Shavlik, "Using symbolic learning to improve knowledge-based neural networks," In AAAI, pp. 177-182, 1992.
[5] G. G Towell and J. W. Shavlik, "Extracting refined rules from knowledge-based neural networks," In Machine learning, vol. 13, no. 1, pp. 71-101, 1993.
[6] C. L. Giles and C. W. Omlin, "Extraction, insertion and refinement of symbolic rules in dynamically driven recurrent neural networks," Connection Science, vol. 5, no. 3-4, pp. 307-337, 1993.
[7] M. Charles and C. Jorgensen, Direct adaptive aircraft control using dynamic cell structure neural networks. NASA Technical Memorandum, Ames Research Center, 1997.
[8] M. Darrah, A. Rubenstein, E. Sorton, and B. DeRoos, "On-board health-state awareness to detect degradation in multirotor systems," In Proceedings of International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1134-1141, 2018.
[9] J. Bruske and G. Sommer, "Dynamic cell structures," Advances in neural information processing systems, pp. 497-504, 1995.
[10] B. Fritzke, "Growing cell structures a self-organizing network for unsupervised and supervised learning," Neural networks, vol. 7, no. 9, pp. 1441-1460, 1994.
[11] T. Martinetz, "Competitive hebbian learning rule forms perfectly topology preserving maps," In ICANN'93, pp. 427-434, 1993.
[12] Darrah, M. and Taylor, Brian. (2011) Chapter 5: Rule Extraction to Understand Changes in an Adaptive System in Adaptive Control Approach for Software Quality Improvement (W. Eric Wong and Bojan Cukic editors) World Scientific. 115-144.
[13] M. Darrah, B. J. Taylor, and S. T. Skias. "Rule extraction from dynamic cell structure neural network used in a safety critical application," In Proceedings of Florida Artificial Intelligence Research Symposium, Miami, FL, May 2004.
[14] L. L. Pullum, B. J. Taylor, and M. Darrah, Guidance for the Verification and Validation of Neural Networks, vol. 11. John Wiley & Sons, 2007.
[15] B. J. Taylor and M. A. Darrah, "Rule extraction as a formal method for the verification and validation of neural networks." In Proceedings of 2005 IEEE International Joint Conference on Neural Networks, vol. 5, pp. 2915-2920, 2005.
[16] G. Bologna and Y. Hayashi, "A Comparison Study on Rule Extraction from Neural Network Ensembles, Boosted Shallow Trees, and SVMs," Applied Computational Intelligence and Soft Computing, vol. 2018, Article ID 4084850, 20 pages, 2018. https://doi.org/10.1155/2018/4084850.
[17] Y. xX. Liu, F. Doctor, S. Z. Fan, and J. S. Shieh, "Performance Analysis of Extracted Rule-Base Multivariable Type-2 Self-Organizing Fuzzy Logic Controller Applied to Anesthesia," BioMed Research International, vol. 2014, Article ID 379090, 19 pages, 2014. https://doi.org/10.1155/2014/379090.
[18] R. Setiono and H. Liu, "Understanding neural networks via rule extraction," In IJCAI, vol. 1, pp. 480-485, 1995.
[19] S.M. Kamruzzaman and A. R. Hasan, "Rule extraction using artificial neural networks, arXiv, preprint arXiv:1009.4984, 20102010.
[20] M. Darrah, B. J. Taylor, M. Webb, and R. Livingston. "A geometric rule extraction approach used for verification and validation of a safety critical application," in Proceedings of Florida Artificial Intelligence Research Symposium Conference, 2005.