Rule Extraction From Dynamic Cell Structure Neural Networks Used in a Safety Critical Application

Marjorie Darrah, Brian Taylor, Spiro Skias

Institute for Scientific Research, Inc.
Fairmont, WV 26554
mdarrah@isr.us, btaylor@isr.us, sskias@isr.us

Abstract

This paper describes an algorithm to extract rules from a dynamic cell structure (DCS) neural network and the rationale for extracting these rules. The DCS is a form of self-organizing map (SOM) neural network that has been used in a real-time adaptive flight control application. The purpose for extracting rules in this instance is to determine whether such rules, along with other techniques, could be used in the verification and validation (V&V) of a neural network being used in a safety-critical role. This paper will explain the intelligent flight control application of the DCS, describe the method used for rule extraction, provide experimental results of the rule extraction techniques applied to several data sets, and examine the relevance of the rules to the V&V process.

Introduction

Neural networks are members of a class of software well suited for domains of non-linearity and high complexity that are ill defined, unknown, or just too complex for standard programming practices. Verifying correct operation of neural networks within projects such as autonomous mission control agents, vehicle health monitoring systems, adaptive flight controllers, or nuclear engineering applications requires a rigorous approach.

Testing the neural network with similar data like that used in training is one of a few methods used to verify that a network has adequately learned the input domain. In non-critical applications, such traditional testing techniques prove adequate for the acceptance of a neural network system that has been trained using input conditions that do not vary from operational conditions. However, in more complex, safety- and mission-critical systems, the standard neural network training-testing approach alone is not able to provide a reliable method for their certification.

The V&V challenge is further compounded by adaptive neural network systems that modify themselves, or "learn," during operation. Traditional software assurance methods fail to account for systems that change after deployment.

We have investigated a set of techniques known as neural network rule extraction to determine their usefulness toward the V&V of neural networks in safety-critical applications. Neural network rule extraction is a technique that translates the decision process of a trained neural network into an equivalent decision process represented as a set of rules. The technique of rule extraction has been used to model the knowledge that the neural network has gained while training or adapting. The rules extracted are generally represented by a set of if-then statements that may be examined by a human. If the neural network is fixed after training then the rules should, with some confidence level, model the way the neural network will handle other data that is processed. If the neural network is an online adaptive neural network, then rule extraction can be done for one point of time to establish what the system looks like at that instance. Repeated application of rule extraction could yield an understanding of the progression of the network during adaptation.

There are many researchers investigating the area of neural network rule extraction (Towell and Shavlik 1993, Thrun 1993, Andrew, Diederich, and Tickle 1995, Carpenter and Tan 1995, Wettayaprasit and Lursinsap 2002). The techniques developed thus far are very neural network specific. Two specific rule extraction techniques seemed closely related to our work. One was used to extract rules from a local cluster neural network and the other from a radial basis neural network; both are similar to the DCS.

One technique, RULEX, was applied first to a constrained error backpropagation multilayer perceptron (Andrews and Geva 1995) and then to a local cluster neural network (Andrews and Geva 2002). Another technique related to our effort is LREX, used to extract rules from radial basis function (RBF) neural networks (McGarry, Wermter, and Macintyre 1999, 2001). Although these approaches seemed to be related to our task of extracting rules from the DCS, both techniques were too specific to be directly applied. Therefore, a new technique had to be developed for rule extraction from the DCS.

Application of Neural Networks to Intelligent Flight Control Systems

The Intelligent Flight Control (IFC) project is working towards developing of a real-time adaptable flight control

system utilizing neural networks (Song et al. 2002). This project is a collaborative effort among the NASA Dryden Flight Research Center (DFRC), the NASA Ames Research Center (ARC), Boeing Phantom Works, the Institute for Scientific Research, Inc. (ISR), and West Virginia University (WVU). The first generation flight control concept (GEN1) was designed to identify aircraft stability and control characteristics using neural networks, and use this information to optimize aircraft performance in both normal and simulated failure conditions.

Two types of neural networks are designed into the IFC GEN1 scheme. A pre-trained, non-adaptive neural network component provides a baseline approximation of stability and control derivatives for the aircraft. The second neural network is an online adaptive network that learns and adapts during flight to account for aerodynamic changes, such as ones due to actuator failures.

The IFC GEN1 system recently completed successful testing in flight on the NASA F-15 Advanced Control Technology for Integrated Vehicles (ACTIVE) aircraft. This aircraft has been highly modified from a standard F-15 configuration to include canard control surfaces, thrust vectoring nozzles, and a digital fly-by-wire flight control system to enable the simulation of different actuator failures during flight.

The Dynamic Cell Structure Neural Network

The online adaptive neural network, the DCS, used in the GEN1 system is of special concern with respect to V&V. The DCS, is a member of a group of neural networks known as self-organizing maps (SOMs). The DCS algorithm, implemented in the GEN1 system by NASA ARC (Jorgensen 1997), was originally developed by (Bruske and Sommer 1994) and is a derivative of work by (Fritzke 1994) combined with competitive Hebbian learning by (Martinez 1993). These neural networks are designed as topology representing networks whose roles are to learn the topology of an input space with perfect preservation.

The DCS neural network learns the function that describes a map of the input space, represented as Voronoi regions. The neurons within the neural network represent the reference vector (centroid) for each of the Voronoi regions. The connections between the neurons, c_{ij}, are then part of the Delaunay triangulation connecting neighboring Voronoi regions through their reference vectors.

This reference vector is known as the "best matching unit" (BMU). Given an input, v, the BMU is the neuron whose weights, w, are closest to v. Along with the BMU, the second BMU (SEC) is found to maintain the Delaunay triangulation and to adjust nearby neurons within the BMU neighborhood, defined as the neurons connected to the BMU through the triangulation.

The DCS algorithm consists of two learning rules, Hebbian and Kohonen. Hebbian learning updates c_{ij} between neurons i and j to reflect the topology (triangulation) of the input space:

$$\dot{c}_{ij} = \begin{cases} 1 & i \in [BMU, SEC],\ j \in [BMU, SEC] \\ \alpha \cdot c_{ij} & \alpha \cdot c_{ij} > \theta \\ 0 & \alpha \cdot c_{ij} < \theta \\ 0 & i = j \end{cases}$$

where the connection is a perfect fit of 1, if i and j are the BMU and SEC. The forgetting constant, α, is included to produce a weakening between i and j if they are not currently the closest to the stimulus, and θ is the edge threshold, a minimum acceptable connection strength in order for the connection to be considered valid. Kohonen learning is used to adjust the weight vectors, w, of the neurons. The change in the weight vectors is represented by:

$$\Delta w_{BMU} = \varepsilon_{BMU}(v - w_{BMU})$$
$$\Delta w_j = \varepsilon_N (v - w_j)$$

where ε_{BMU} is the BMU weight adjustment parameter and ε_N is the weight adjustment applied to the neighborhood of the BMU.

These two learning rules allow the DCS neural network to change its structure. The ability to add new neurons into the network as it grows gives the DCS neural network the potential to evolve into many different configurations. This adaptive nature could open up the possibility of sub-optimal or even erroneous solutions.

Developers of the DCS neural network have been cautious about expanding their use into safety- and mission-critical domains due to the complexities and uncertainties associated with these complex, adaptive software systems. Since the DCS neural network and other adaptive neural networks are beginning to be used within high-assurance systems, NASA has encouraged research in the area of V&V of neural networks to answer the question: How can we be sure that any system that includes neural network technology is going to behave in a known, consistent and correct manner?

The Rule Extraction Technique

As mentioned in the introduction it was our objective to examine rule extraction as a possible way to V&V a critical system containing a neural network.

The algorithm developed for extracting rules from the DCS is a modification of the LREX algorithm by McGarry et al. that was used to extract rules from a RBF neural network (McGarry, Wermter, and Macintyre 1999, 2001). The weights of the DCS after it has been trained are used as inputs to the algorithm because they form the centers of the Voronoi regions. The BMU corresponding to each data point is recorded during training and also used as input to the algorithm. The training data is divided into regions

based on the BMU. Then for each region x_{lower} is the smallest value of the independent variable that has a particular BMU and x_{upper} is the largest value of that independent variable that has that same particular BMU. These two numbers form bounds for the intervals in the antecedent. (Example: (variable >= x_{lower} AND <= x_{upper})) An interval is determined for each of the independent variables and the statements are connected by AND to form the full antecedent. When the DCS was used as a classifier with the Iris data the conclusion of the if-then statement was categorical. In this case the category associated with the BMU was reported in the rule as the conclusion. When the DCS was used to learn a function and the dependent variables were continuous then the conclusion was stated the same way the antecedent was stated, intervals connected with ANDs. The algorithm used for the rule extraction is in Figure 1.

Input:
Weights of the DCS (centers of Voronoi region)
Best matching unit for each input
Output:
One rule for each cell of the DCS
Procedure:
Train DCS on data set
Record BMU for each input
Collect all inputs with common BMU to form cell
For each weight (w_i)
    For each independent variable
        $x_{lower} = \min\{x \mid x \text{ has BMU} = w_i\}$
        $x_{upper} = \max\{x \mid x \text{ has BMU} = w_i\}$
Build rule by:
    Independent variable in [x_{lower} , x_{upper}]
    Join antecedent statements with AND
    Dependent variable = category
        OR
    Dependent variable in [y_{lower} , y_{upper}]
    Join conclusion statements with AND
Write Rule

Figure 1 Rule Extraction Algorithm

Experimental Results

The testing of the extracted rules consisted of a multi-phase approach: data separation and randomization, training the DCS, extract the rules from the DCS, implement the extracted rules, test the rules with a test data set, and compare the results to the DCS (Figure 2)

The initial testing of the algorithm was done using the benchmark Iris data set to obtain results that could be compared to other author's results (Fisher 1936). Eventually, the Iris data would be replaced with IFC flight data. A five-fold cross validation approach was used for the testing of the Iris data. This meant that the Iris data was divided into five equal parts, $\{S_1, S_2, S_3, S_4, S_5\}$, of which four parts were used for training and a fifth part is used for testing. The Iris data contains 150 data points; therefore, the data was partitioned into five groups of 30 random points.

After training on the Iris data, the DCS should have clustered the input data into different classifications representing the different Iris types: Setosa, Virginica, or Versicolor. To capture what the DCS learned, the rule extraction algorithm was applied to the DCS. The output was a set of rules in the form of if-then statements. These rules represent the Voronoi regions that the DCS formed to cluster the Iris data. An example rule for the Iris Data looks like:

RULE FOR CELL 1
IF (SL >=5.6 AND <=7.9) AND
(SW >=2.2 AND <=3.8) AND
(PL >=4.8 AND <=6.9) AND
(PW >=1.4 AND <=2.5)
THEN Virginica

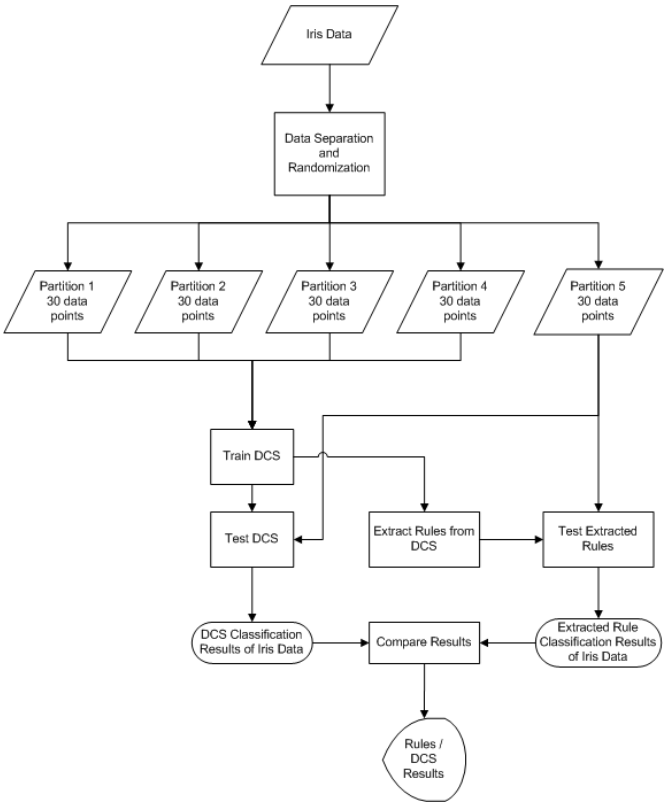

Figure 2 Testing Approach Block Diagram

Since the extracted rules are a representation of the DCS, they should classify data in the same way that the DCS classifies data. To test the rules, they were

implemented in MATLAB and applied to the test subset of the Iris data. The results from the extracted rules were then compared to the results from the same data classified by the DCS.

Five iterations of this procedure were completed to ensure that each subset was used as test data. This meant that iteration one used subsets $\{S_1, S_2, S_3, S_4\}$ to train and S_5 to test, iteration two used partitions $\{S_1, S_2, S_3, S_5\}$ to train and S_4 to test, and so on.

After both the trained DCS and the extracted rules were applied to the test set. The results were compared for all five iterations. The DCS output and the extracted rules output had an overall agreement of 82% in classifying the Iris data.

The next step was to train the DCS on IFC flight data. The DCS was used for the purpose of classification with the Iris data, however with the flight data the DCS is used to learn a function. The flight data used was obtained from an F-15 Flight Simulator developed at WVU for use in testing the IFC GEN1 scheme (Perhinschi et al. 2002). This data set contains seven independent variables and 26 dependent variables. These variables were introduced to one of five different DCS networks, one network for each of the aerodynamic derivative coefficients: C_z, C_m, C_l, C_n, and C_y. Each network learns the derivatives associated with a different coefficient. For example, C_m learns the stability and control derivatives associated with pitching moments due to normal force and uses the inputs of mach, altitude, alpha, and beta as the independent variables and cma, cmdc, cmds and cmq as the dependent variables.

After training the DCS on these variables the rules extracted take on the form seen in Figure 3. The data in this case is continuous data so the rules give both antecedent and conclusion in the form of intervals.

Using Extracted Rules for V&V of the DCS

The objective of our overall research initiative is to develop methods or techniques that can assist in the V&V of neural networks (ISR 2002). The extraction of rules is one such method. After extracting the rules from the IFC neural networks, the rules were compared against the two documents provided by the IFC project team, the Software and Interface Requirements Specification (SIRS) (ISR 2001a), and the Interface Control Document (ICD), (ISR 2001b). The usefulness in requirements traceability and the overall usefulness of rule extraction to the IV&V practitioners understanding were assessed.

The first way the extracted rules were used is for **comparison directly against requirements**. In the IFC case the rules can be used specifically for assessing the expected ranges of the inputs and outputs of the DCS system. For V&V purposes a comparison can be made between the ranges from the SIRS document and the ranges in the antecedents of the rules.

This verification activity proves to be worthwhile since there are certain situations that may lead to the DCS network rule antecedents that violate the input range limits.

```
RULES FOR CELL1
        IF (mach >=0.73926 AND <=0.7738) AND
        (altitude >=20271.1609 AND <=21233.6014) AND
        (alpha >=2.1508 AND <=2.4619) AND
        (beta >=-0.079409 AND <=0.0035506)
        THEN (cma >=0.34593 AND <=0.40947) AND
        (cmdc >=-0.0025557 AND <=0.0029545) AND
        (cmds >=0 AND <=0) AND
        (cmq >=0 AND <=0)
RULES FOR CELL2
        IF (mach >=0.77404 AND <=0.78946) AND
        (altitude >=19860.1718 AND <=20264.4617) AND
        (alpha >=1.8323 AND <=2.1414) AND
        (beta >=-0.021984 AND <=0.020729)
        THEN (cma >=0.40578 AND <=0.47844) AND
        (cmdc >=0.0031626 AND <=0.0079892) AND
        (cmds >=0 AND <=0) AND
        (cmq >=0 AND <=0)
RULES FOR CELL3
        IF (mach >=0.78455 AND <=0.8178) AND
        (altitude >=19205.5546 AND <=19999.3379) AND
        (alpha >=1.2545 AND <=1.8311) AND
        (beta >=-0.20803 AND <=0.64913)
        THEN (cma >=0.274 AND <=0.40517) AND
        (cmdc >=0.0080297 AND <=0.016743) AND
        (cmds >=0 AND <=0) AND
        (cmq >=0 AND <=0)
```

Figure 3 Rules Extracted from DCS Trained on Flight Data

One possibility could be the improper use of a DCS "initialization" point. If the DCS networks start with no knowledge and are initialized at an improper starting point (say the first two neurons within DCS are initialized to values of zero, something not unreasonable), then these initial values may affect the Voronoi regions within the DCS in an adverse way causing them to allow values outside the flight envelope. Looking at the antecedents of the extracted rules identifies this situation.

The rules may be also be used for analysis of specific input variables. For example, mach and altitude are inputs that do not cross the zero value within their acceptable ranges. One may want to check to ensure that the extracted rules align with the expected ranges, and in situations where the zero value is within a rule, perhaps identify this as an area of special interest that requires additional analysis.

Another way the rules can be used is for **comparison against implied design considerations**. The V&V task of requirements traceability throughout the lifecycle is an important step. However in innovative research applications, such as the IFC project, the nature of the research may mean that well-defined requirements are hard to elicit. Instead the project makes use of higher-level

specifications that are hard to direct against the neural networks. Adaptive neural networks may make the requirements generation even more difficult. Since the extracted rules represent the learning of the neural network, the rules can be used to determine if the network is meeting these higher-level specifications when the requirements are not written clearly or concisely, yet the overall goal for the system is generally well understood.

For example, with regards to flight control, one requirement for the network may be smooth transitions of derivatives between regions within the flight envelope. If the neural network has been trained to learn these derivatives, one way to know whether the network has learned properly is to examine network knowledge, via rule-extraction, near the boundaries of the flight envelope regions. The rules allow for an inspection of how the antecedents and rule conclusion change and if the ranges over which they change are an indication of the smooth transition. If system designers are not satisfied with the network performance, then it is possible the network may require further training or modification. Either way, the examination of the extracted rules can help to map back neural network knowledge to system requirements.

The rules may have an additional use in examining different modes of operation. The purpose of the IFC system is to induce "safe" failures and allow the system to adapt to accommodate these failures. Some of the failures may induce learning that extends beyond the expected ranges of the inputs. Retaining prior learning, especially after a failure occurs, may be something interesting the IFC project team will want to investigate. The rules may identify Voronoi regions that violate the expected input ranges and point to an area of the input space that will require further simulation and understanding before the project proceeds.

Conclusion

The goal of this research is to demonstrate that rules extracted from the DCS neural network could be used to assist in the V&V of the neural network in a safety-critical application. The rules are viewed as a descriptive representation of how the DCS "handles" data. This representation provides the inner knowledge of the neural network that can be used to help understand whether the neural network is functioning as expected.

A V&V practitioner might use rule extraction for different purposes. They could apply rule extraction to obtain a set of rules that mimics the functionality of the network and then use these rules for comparison against the original set of requirements. This would provide information for review of the correctness of the function the network is approximating. Additionally, these rules could be validated through the use of formal methods, such as a model checker. At a minimum, extraction of these rules would provide some sense of confidence that the network will behave as it was intended.

Methods for determining the accuracy of the rules is also still under investigation. In the case when the DCS is being used as a classifier with the Iris data, it was easy to access the accuracy of the rules by simply comparing the classification results. Determining the accuracy of the rules when the DCS is trained with flight data is not as straight forward. We will attempt to evaluate the rules by first looking at the domain coverage and then by looking at the actual difference between the rule boundaries and the Voronoi region boundaries.

We are still refining the algorithm for the DCS rule extraction. Our current algorithm is a simplistic attempt to capture the Voronoi regions that are created by the DCS. For future refinement we will attempt to better capture the n-dimensional convex hulls that make up these Voronoi regions with the rules. The problem in doing so is that the explicit description of such regions may become overly complicated and no more understandable than the current information that can be obtained by looking at the weights and other parameters of the actual neural network. There is a definite trade off between accuracy and understandability. We must ensure that any rules extracted are accurate, useful, and understandable. This is the challenge ahead as we continue to investigate this topic.

We propose to continue our research in the area by developing the neural network rule extraction technology to a level where it can be transferred into a software tool that could facilitate the V&V. The feasibility study for this continuing effort is funded by NASA Ames Research Center under an upcoming STTR award with Prologic, Inc., Fairmont, WV.

Acknowledgements

This research was sponsored by NASA, under Research Grant NAG5-12069.

References

Towell, G and J. Shavlik. 1993. The extraction of refined rules from knowledge based neural networks. *Machine Learning* 13(1):71-101.

Thrun, Sebastian B. 1993. Extracting Provably Correct Rules from Artificial Neural Networks, Technical Report IAI-TR-93-5, Institute fur Informatics III, Universität Bonn.

Andrews, Robert; J. Diederich; and A. B. Tickle. 1995. A Survey and Critique Of Techniques For Extracting Rules From Trained Artificial Neural Networks. *Knowledge Based Systems* 8:373-389.

Carpenter, A and A.H. Tan. Rule extraction: from neural architecture to symbolic representation. *Connection Science* 7(1):3-27.

Wettayaprasit, Wiphada and Chidchanok Lursinsap. 2002. Neural Rule Extraction based on Activation Projection with Certainty Factor Refinement. *In Proceedings of IEEE World Congress on Computational Intelligence*.

Andrews, R. and S.Geva. 1995. RULEX & CEBP networks as the basis for a rule refinement system. *In Hybrid Problems, Hybrid Solutions*, ed. John Hallam. IOS Press.1-12

Andrews, R and S. Geva. 2002. Rule Extraction From Local Cluster Neural Nets. *Neurocomputing* 47:1-20.

McGarry, Kenneth, John Tait, Stefan Wermter, and John McIntyre. 1999. Rule-Extraction from Radial Basis Function Networks. *In Proceedings of International Conference on Artificial Neural Networks* 1:613-618. Edinburgh, Scotland.

McGarry, Kenneth, Stefan Wermter and John Macintyre. 2001. The Extraction and Comparison of Knowledge from Local Function Networks, *International Journal of Computational Intelligence and Applications* 1(3): 369-382.

Fisher, A. 1936. *Annals of Eugenics* 7:179-188.

Song, Yongkyu, Giampiero Campa, Marcello R. Napolitano, Brad Seanor, Mario G. Perhinschi. 2002. On-Line Parameter Estimation Techniques Comparison Within a Fault Tolerant Flight Control System. *AIAA Journal of Guidance, Control, and Dynamics* 25(3):528-537.

Jorgensen, Charles C. 1997. Direct Adaptive Aircraft Control Using Dynamic Cell Structure Neural Networks. NASA Technical Memorandum 112198, NASA Ames Research Center.

Bruske, Jorg and Gerald Sommer. 1994. Dynamic Cell Structures. *In Proceedings of Neural Information Processing Systems (NIPS)*, 497-504.

Fritzke, B. 1994. Growing Cell-Structures – a Self-Organizing Network for Unsupervised and Supervised Learning, *Neural Networks*, 7(9): 1441-1460.

Martinetz, T. M. 1993. Competitive Hebbian Learning Rule Forms Perfectly Topology Preserving Maps. In *Proceedings of International Conference on Artificial Neural Networks* (ICANN) 427-434. Amsterdam: Springer.

Perhinschi, M. G., G. Campa, M. R. Napolitano, M. Lando, L. Massotti, and M.L. Fravolini. Modeling and Simulation of a Closed Loop Aircraft/Fault Tolerant Control System. Submitted to*: International Journal of Modeling and Simulation.*

Institute for Scientific Research, Inc. (ISR). 2002. Toward Reliable Neural Network Software for The Development of Methodologies for the Independent Verification of Neural Networks. IVVNN-LITREV-F001-UNCLASS-11120

Institute for Scientific Research, Inc. (ISR). 2001. Software and Interface Requirements Document (SIRS). IFC-SIRS-F004-UNCLASS-051501

Institute for Scientific Research, Inc. (ISR). 2001. Interface Control Document (ICD). IFC-ICD-F008-UNCLASS-01150.