Performance Estimation of a Neural Network-based Controller

Johann Schumann[1] and Yan Liu[2]

[1] RIACS / NASA Ames, Moffett Field, CA 94035, schumann@email.arc.nasa.gov
[2] Motorola Labs, Schaumburg, IL 60193, yanliu@motorola.com

Abstract. Biologically inspired soft computing paradigms such as neural networks are popular learning models adopted in adaptive control systems for their ability to cope with a changing environment. However, continual changes induce uncertainty that limits the applicability of conventional validation techniques to assure a reliable system performance. In this paper, we present a dynamic approach to estimate the performance of two types of neural networks employed in an adaptive flight controller: the validity index for the outputs of a Dynamic Cell Structure (DCS) network and confidence levels for the outputs of a Sigma-Pi (or MLP) network. Both tools provide statistical inference of the neural network predictions and an estimate of the current performance of the network. We further evaluate how the quality of each parameter of the network (e.g., weight) influences the output of the network by defining a metric for parameter sensitivity and parameter confidence for DCS and Sigma-Pi networks. Experimental results on the NASA F-15 flight control system demonstrate that our techniques effectively evaluate the network performance and provide validation inferences in a real-time manner.

1 Introduction

Adaptive Flight Control is considered as one of the most challenging real-time automation and control tasks as the system's functions are not static but evolve over time in a non-probabilistic manner. While these evolving functions, through judicious online learning, aid the adaptive controller to recuperate the system (aircraft) from an operational damage (sensor/actuator failure, changed aircraft dynamics: broken aileron or stabilator, etc.), they add an additional degree of complexity and system uncertainty. Since it is impossible to estimate and analyze all possible concerns relative to system safety beforehand, online adaptive systems require a non-conventional approach to verification and validation (V&V).

Neural networks are widely employed for function approximation, prediction and pattern recognition. The requirements on such models are usually described as satisfying certain criteria of precision and/or accuracy. Typical metrics used for performance evaluation of neural networks are Mean Square Error (MSE), Squared Error, etc. They are used to measure the learning performance of a neural network model. For prediction performance evaluation, the most popular metrics are prediction/confidence intervals defined to measure the reliability

of network output. In the context of an online neural network based adaptive control system, the online neural network is expected to promptly respond to, adapt to and accommodate environmental changes. Therefore, within an online adaptive system, assuring the performance of the online neural network requires online evaluation of its adaptation performance.

In this paper, we present statistical approaches to estimate the performance of a neural network: the validity index (VI) for Dynamic Cell Structures (DCS) and confidence levels for the outputs of a Sigma-Pi (or MLP) network. Both tools provide error bars of the neural network outputs, given the current inputs and thus provide an estimate of the current performance of the network.

For safety-critical systems, a "black box" approach to network assessment is not sufficient. We therefore estimate how the quality of each parameter of the network (e.g., weight) influences the output of the network by calculating parameter sensitivity and parameters confidence for DCS and Sigma Pi networks.

There is only little related work on verification and validation of neural network-based controllers. [7, 18] discuss an approach toward V&V of such systems; NASA has developed a software verification process guide [12] addressing these issues. The use of Bayesian techniques to estimate neural network quality is presented in detail in [2]; another metric (validity index) for RBF has been introduced in [9]. Monitoring approaches for neuro-adaptive controllers, based on Lyapunov stability are discussed in [6].

2 Neural Network based Flight Control

We illustrate our approach with the NASA F-15 Intelligent Flight Control System (IFCS) project. Its aim is to develop and test-fly a neuro-adaptive intelligent flight control system for a manned F-15 aircraft. Two principal architectures have been developed: the Gen-I architecture uses a DCS neural network as its online adaptive component, the Gen-II architecture a Sigma Pi network.

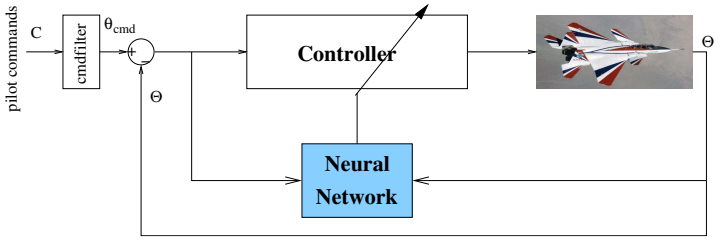

Fig. 1. IFCS Generic Adaptive Control Architecture

Figure 1 shows the basic architecture of the Gen-I and Gen-II controllers: pilot stick commands Θ_{cmd} are mixed with the current sensor readings Θ (e.g., airspeed, angle of attack, altitude) to form the desired behavior of the aircraft. From these data, the PID controller calculates the necessary movements of the control surfaces (e.g., rudder, ailerons) and commands the actuators. The controller incorporates a model of the aircraft dynamics. If the aerodynamics of the

aircraft changes (e.g., due to a damaged wing or a stuck rudder), there is a deviation between desired and actual state. The neural network is trained during operation to minimize this deviation. Whereas in the Gen-I architecture, the appropriate control derivatives are modified with a neural network, Gen-II uses a dynamic inverse controller with control augmentation, i.e., the neural network produces a control correction signal. For details on the control architecture see [16, 4, 17].

2.1 The Neural Networks

Dynamic Cell Structure Network The Dynamic Cell Structures network is derived as a dynamically growing structure in order to achieve better adaptability. DCS can be seen as a special case of Self-Organizing Map (SOM) structures as introduced by Kohonen [8] and further improved to offer topology-preserving adaptive learning capabilities that can respond and learn to abstract from a much wider variety of complex data manifolds [13, 3]. In the IFCS Gen-I controller, the DCS provides derivative corrections during system operation.

The DCS network adopts the self-organizing structure and dynamically evolves with respect to the learning data. It approximates the function that maps the input to the output space. At last, the input space is divided into different regions, referred to as the Voronoi regions [13, 3, 5]. Each Voronoi region is represented by its centroid, a neuron associated with its reference vector known as the "best matching unit" (bmu). Further, a "second best matching unit" (sbu) is defined as the neuron whose reference vector is the second closest to a particular input. An Euclidean distance metric is adopted for finding both units.

The training algorithm of the DCS network combines the competitive Hebbian learning rule and the Kohonen learning rule. The Hebbian learning rule is used to adjust the connection strength C_{ij} between two neurons. The Kohonen learning rule is used to adjust the weight representations of the neurons ($\boldsymbol{w_i}$), which are activated based on the best-matching methods during the learning. If needed, new neurons are inserted. After learning, when DCS is used for prediction (the recall mode), it will recall parameter values at any chosen dimension. It should be noted that the computation of an output is different from that during training. When DCS is in recall mode, the output is computed based on two neurons for a particular input. One is the bmu of the input; the other is the closest neighbor of the bmu other than the sbu of the input. In the absence of neighboring neurons of the bmu, the output value is calculated using the bmu only. Since our performance estimation does not depend on the specific learning algorithm, it will not be discussed in this paper. For details on DCS and the learning algorithm see [13, 3, 5, 10].

Sigma Pi Neural Network The IFCS Gen-II controller uses a Sigma-Pi ($\Sigma\Pi$) neural network [15], where the inputs are $\mathbf{x}$ subjected to arbitrary basis functions (e.g., square, scaling, logistic function). Then Cartesian products (Π) of these function values are calculated. The final output of the network o is a weighted

sum (Σ) of these products (Figure 2):

$$o = \sum_i w_i p_i \quad \text{where} \quad p_i = \prod_j \beta(\mathbf{x}_j)$$

with weights w_i and $\beta(\mathbf{x}_j)$ the basis functions. During the training, the weights w_i are modified as to minimize the tracking error of the controller. As our approach to confidence and sensitivity analysis does not depend on the specific training algorithm for the network, it will not be discussed here. For details see again [16]. Figure 2 (right) shows how the network weights w_i develop over time during an operational scenario. At $t = 1.5s$, a failure occurs, triggering adaptation of the neural network.

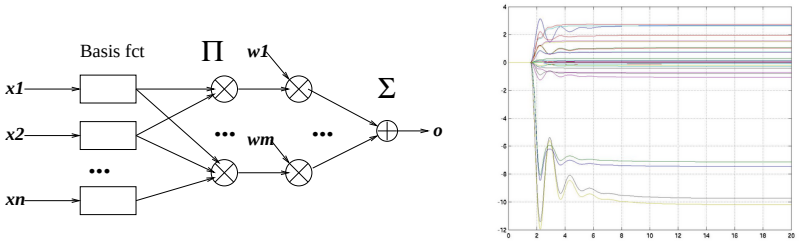

Fig. 2. Architecture of $\Sigma\Pi$ network (left). Development of the NN weights over time during adaptation (right). The failure occurred at $t = 1.5s$.

3 Estimating Network Performance

3.1 Data Confidence

Validity Index Following the definition of Validity Index (VI) in RBF networks by Leonard et.al.[9], we define the validity index in DCS networks as an estimated confidence measure of a DCS output, given the current input. The VI can be used to measure the accuracy of the DCS network fitting and thus provide inferences for future validation activities. Based on the primary rules of DCS learning and properties of the network structure, we employ confidence intervals and variances to calculate the validity index in DCS. The computation of a validity index for a given input consists of two steps: (1) compute the local error associated with each neuron, and (2) estimate the standard error of the DCS output for the given input using information from step (1). Details can be found in [11, 10].

We have modified the DCS training algorithm to calculate the validity index. Because all needed information is present at the final step of each training cycle, we can simply calculate $s_i'^2$ for each neuron after the learning stops. When the DCS is in recall mode, the validity index is computed based on the local errors and then associated with every DCS output. We have simulated the online learning of the DCS network under a failure mode condition. Running at 20 Hz, the DCS network updates its learning data buffer (of size 200) at every second and learns on the up-to-date data set of size 200. We first start the DCS network

under nominal flight conditions with 200 data points. After that, every second, we set the DCS network in recall mode and calculate the derivative corrections for the freshly generated 20 data points, as well as their validity index. Then we set the DCS network back to the learning mode and update the data buffer to contain the new data points.

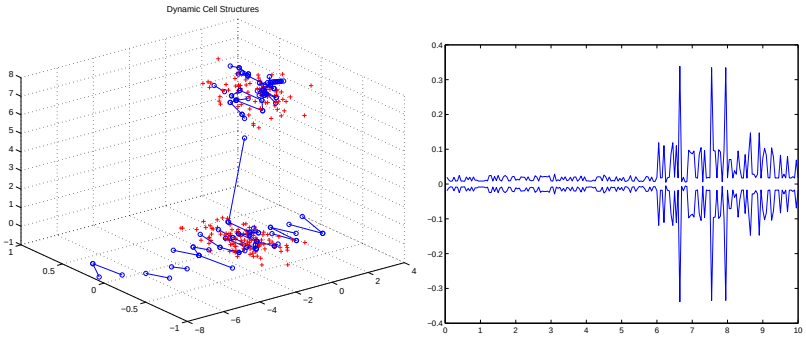

Fig. 3. Online operation of DCS VI on failure mode simulation data. Left: The final form of DCS network structures. Right: VI shown as error bars for each DCS output.

Figure 3 shows the experimental results of our simulation on the failure mode condition. The left plot shows the final form of the DCS network structure at the end of the simulation. The 200 data points in the data buffer at the end of the simulation are shown as crosses in the 3-D space. The network structure is represented by circles (as neurons) connected by lines as a topological mapping to the learning data. The right plot presents the validity index, shown as error bars. The x-axis here represents the time frames. The failure occurs at $t = 5.0s$. We compute the validity index for the data points that are generated five seconds before and five seconds after the failure occurs.

A trend revealed by the validity index in our simulations is the increasingly larger error bars after the failure occurs. At $t = 6.0s$, the network has learned these 20 failure data points generated from $\Delta t = 5.0 \sim 6.0s$. The network performance became less stable. After that, the error bars start shrinking while the DCS network adapts to the new domain and accommodates the failure. After the failure occurs, the change (increase/decrease) of the validity index varies depending on the characteristics of the failure as well as the accommodation performance of the DCS network. Nevertheless, the validity index explicitly indicates how well and how fast the DCS network accommodates the failure.

Bayesian Confidence Tool For the Gen-II architecture, the *Confidence Tool* (CT) [7] produces a quality measure of the neural network output. Our performance measure is the probability density $p(o|\mathbf{x}, D)$ of the network output o given inputs $\mathbf{x}$, when the network has been trained with training data D. Assuming a Gaussian distribution, we use the standard deviation σ^2 as our performance measure. A small σ^2 (a narrow bell-shaped curve) means that, with a

high probability, the actual value is close to the returned value. This indicates a good performance of the network. A large σ^2 corresponds to a shallow and wide curve. Here, a large deviation is probable, indicating poor performance.

Our confidence tool uses an algorithm, following the derivation in [2]. The CT has been implemented for Sigma-Pi and multi-layer perceptron (MLP) networks in Matlab (for a Simulink environment) and in C. For details see [7, 18]. The CT tool is currently implemented on the flight computer on a NASA F-15 aircraft and is scheduled for manned test-flights in the near future.

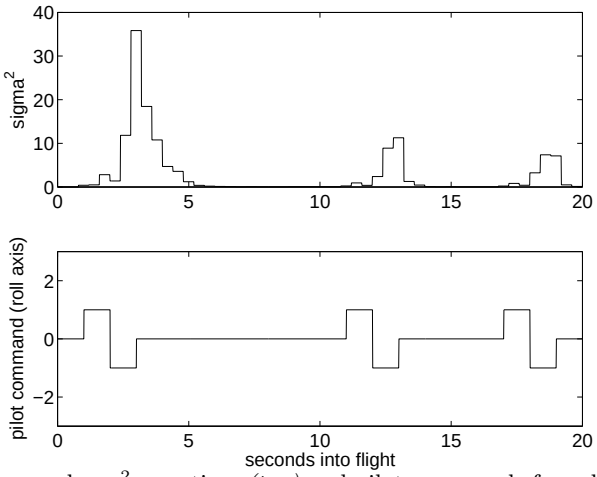

Fig. 4. Confidence value σ^2 over time *(top)* and pilot commands for roll axis (*bottom*). A failure has occurred at $t = 1.5s$.

Figure 4 shows the results of a (Simulink) simulation experiment. In the top panel, σ^2 is shown over time. At time $t = 1.0s$, the pilot issues a doublet command (fast stick movement from neutral into positive, then negative and back to neutral position; Fig. 4*(lower panel)*). Shortly afterwards ($t = 1.5s$), one control surface of the aircraft (stabilizer) gets stuck at a fixed angle ("the failure"). Because the system dynamics and the model behavior do not match any more, the neural network has to produce an augmentation control signal to compensate for this deviation. The σ^2 of the network output increases substantially, indicating a large uncertainty in the network output. Due to the online training of the network, this uncertainty decreases very quickly.

A second and third pilot command (identical to the first one) is executed at $t = 11s$, and $t = 17s$, respectively. During that time, the network's confidence is still reduced, but much less than before. This is a clear indication that the network has successfully adapted to handle this failure situation.

3.2 Sensitivity and Confidence

For the analysis of any controller's behavior, it is important to estimate its sensitivity with respect to input perturbations. A badly designed controller might

amplify the perturbations, which could lead to oscillations and instability. The higher the *robustness* of the controller, the less influence arises from input perturbations. It is obvious that such a metric (i.e., $\frac{\partial \mathbf{o}}{\partial \mathbf{x}}$ for outputs $\mathbf{o}$ and inputs $\mathbf{x}$) is also applicable to an adaptive control system. For an adaptive component, like a neural network, the estimation of the sensitivity is a "black box" method, i.e., no knowledge about the internal structure or parameters is necessary.

In this paper, however, we focus on *parameter sensitivity*. This means, we calculate $\frac{\partial \mathbf{o}}{\partial p}$ for each of the adjustable parameters $p \in \mathcal{P}$. For a neural network, $\mathcal{P}$ is comprised of the network weights w_i, for the DCS network, it is the reference vectors of the neurons $\boldsymbol{w_i}$. During training of the network, these parameters are adjusted to minimize the error. Depending on the architecture of the adaptive controller, the network can be pre-trained, i.e., the parameters are determined during the design phase ("system identification"), or the parameters are changing while the system is in operation ("online adaptation"). In both cases, one needs to know, which influence the actual values of the parameters have on the output of the neural network: if the influence of a parameter or neuron is negligible, then this neuron might be removed from the network. On the other hand, extremely high sensitivity might cause numerical problems.

Even more information can be obtained if we consider each parameter of the neural network not as a scalar value, but as a probability distribution. Then, we can formulate the sensitivity problem in a statistical way. The probability of the output of the neural network is $p(\mathbf{o}|\mathcal{P}, \mathbf{x})$ given parameters $\mathcal{P}$ and inputs $\mathbf{x}$. If we again assume a Gaussian probability distribution, we can define our parameter confidence as the variance $\sigma^2_{\mathcal{P}}$. In contrast to calculating the network output confidence value, we do not marginalize over the weights, but over the inputs.

A Sensitivity Metric for DCS Networks Within the IFCS Gen-I, the DCS networks are employed for online adaptation/learning. Their parameters (connection strength C_{ij} and reference vectors $\boldsymbol{w}_i$) are updated during system operation. It should be noted that the connection strength C does not contribute to the network predictions while it is in recall mode. This implies that the sensitivity of the connection strength is merely a structure related parameter that influences the reference vectors instead of the network output. We therefore only measure the sensitivity of the reference vector of the DCS network. Using the simulation data obtained from the IFCS Gen-I simulator, we calculate the parameter sensitivity s and its confidence σ^2 after each learning epoch during a flight scenario. The sensitivity analysis has been conducted on a N-dimension space, where N is the number of dimensions of the input space.

Figure 5 shows two sensitivity snapshots at different times of the simulation where the network has been trained with 2-dimensional data. Each neuron is associated with a 2-dimensional sensitivity ellipse. At the beginning of the simulation, the network is initialized with two neurons whose reference vectors represent two randomly selected training data points. The network continues learning and adjusts its own structure to adapt to the data. Figure 5 (left) shows

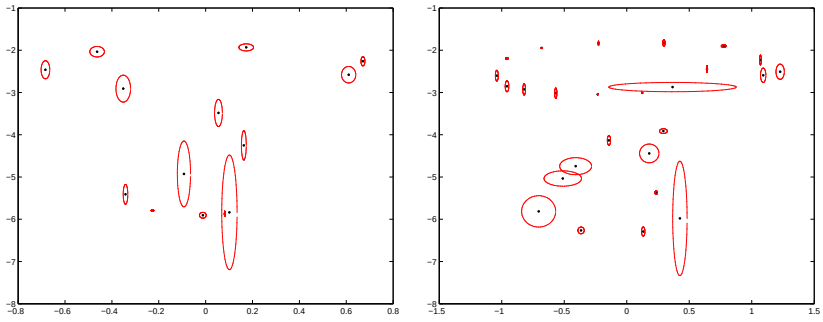

Fig. 5. Sensitivity analysis for DCS networks

the situation at $t = 5.0s$. Figure 5 (right) shows the situation at $t = 10.0s$. At $t = 5.0s$, most neurons exhibit relatively large sensitivity, while only a few (31%) neurons have small sensitivity values. However, at $t = 10.0s$, when the network has well adapted to the data, Figure 5 (right) clearly indicates that now most (78%) neurons have small sensitivity values.

A Sensitivity Metric for Sigma-Pi Networks We have implemented the sensitivity analysis for the online adaptive Sigma-Pi network of the IFCS Gen-II controller. We calculate the parameter sensitivity s and its confidence σ^2 for the network parameters w_i at each point in time during a flight scenario. Figure 6 shows two sensitivity snapshots at various stages of the scenario. At the beginning of the scenario, all parameters of the network are set to zero, giving (trivially) in the same sensitivity. At $t = 1.5$, a failure is induced into the system. In order to compensate for the failure, the network weights adapt (see Fig. 2(right)). Figure 6(left) shows the situation at $t = 5.0s$. A considerable amount of adaptation and weight changes has taken place already. However, the confidence for each of the 60 neurons is still relatively small, as indicated by the large error bars. After approximately 20 seconds, the neural network is fully trained. Figure 6(right) now shows quite different values for the sensitivity. Whereas the sensitivity for most of the neurons is really small now, a few (here 7) neurons exhibit high sensitivity. Although their σ^2 is somewhat larger than that for the other neurons, a clear distinction between the different groups can be made.

Independently from this analysis, the network architecture had been modified several times during the design of the Gen-II controller. So, the number of weights in the network (for roll axis) was reduced from 60 (as shown) to 6. The results obtained with our Parameter Confidence tool (Figure 6) clearly demonstrate (after the fact) that this substantial reduction of the network size is justified.

4 Conclusions

We have presented tools for the estimation of the performance of a neural network used in an adaptive controller. For two, highly disjoint architectures, Dy-

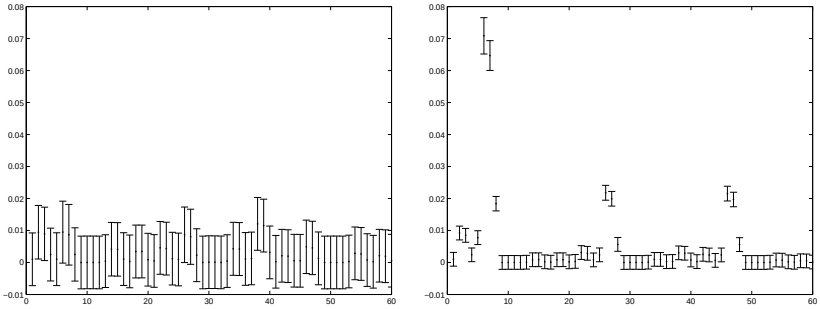

Fig. 6. Parameter sensitivity and confidence at $t = 5s$ (left) and $t = 20s$ (right).

namic Cell Structures (DCS), and Sigma Pi networks, we have shown how the network prediction performance in form of statistical error bars (validity index for DCS, network confidence for Sigma Pi) can be calculated. The online estimation is import in control applications, where the neural network is being trained during operation. The availability of this information plays an important role for verification and validation of such a system in a safety-critical application.

Further insight on the actual performance of the neural network can be gained by looking at the parameter sensitivity and parameter confidence. In this paper, we have presented tools to calculate the sensitivity of individual parameters (reference vectors of neurons in DCS; weights in Sigma Pi networks) and the parameters confidence (again an error-bar). Our approaches are based upon Bayesian statistics and thus provide a solid statistical background for the performance estimation.

Our tools are primarily designed to provide dynamic data on the performance of the networks, but can also be used during the early design phase of an adaptive controller, when the architecture and size of the network is determined. Our Bayesian approach allows different models (e.g. networks with different numbers of hidden units, or different network types such as multi-layer perceptrons, Sigma-Pi, RBF, or DCS) to be compared using only the training data. More generally, the Bayesian approach provides an objective and principled framework for dealing with the issues of model complexity.

Our tools are capable of calculating a performance index for the neural network. The actual performance of the entire system (in our case, the aircraft) also depends on a multitude of other parameters (e.g., robustness of controller, performance metric, type of failure). In aeronautics, the performance of an aircraft is defined in terms of its handling quality (e.g., the Cooper-Harper rating). Current research aims to relate our performance metric with the aircraft handling quality. With the real-time availability of handling quality estimates, our tools can be used to alert the pilot and provide assistance/support to decision making.

References

1. Ahrns, I., Bruske, J., Sommer, G.: On-line Learning with Dynamic Cell Structures. In: Fogelman-Soulié, F., Gallinari, P. (eds.): Proc. Int. Conf. Artificial Neural Networks, Vol. 2. EC2, Nanterre, France (1995) 141-146
2. Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford, UK (1995)
3. Bruske, J., Sommer, G.: Dynamic Cell Structures. In: Proc. Neural Information Processing Systems, Vol. 7. (1995) 497-504
4. Calise A., Rysdyk. R.: Nonlinear Adaptive Flight Control Using Neural Networks. IEEE Control Systems Magazine 21(6) (1998) 14-26
5. Fritzke, B.: Growing Cell Structures - a Self-organizing Network for Unsupervised and Supervised Learning. Neural Networks 7(9) (1993) 1441-1460
6. Fuller, E., Yerramalla, S., Cukic, B.,Gururajan, S.: An Approach to Predicting Non-deterministic Neural Network Behavior. In: Proc. Intl. Joint Conference on Neural Networks (IJCNN) (2005)
7. Gupta, P., Schumann, J.: A Tool for Verification and Validation of Neural Network Based Adaptive Controllers for High Assurance Systems. In: Proc. High Assurance Software Engineering, IEEE Press (2004)
8. Kohonen, T.: Self-Organizing Maps. Springer-Verlag, New York (1997)
9. Leonard, J.A., Kramer, M.A., Ungar, L.H.: Using Radial Basis Functions to Approximate a Function and Its Error Bounds. IEEE Transactions on Neural Networks 3(4) (1992) 624-627
10. Liu, Y.: Validating A Neural Network-based Online Adaptive System. PhD thesis, West Virginia University, Morgantown (2005)
11. Liu, Y., Cukic, B., Jiang, M., Xu, Z.: Predicting with Confidence - An Improved Dynamic Cell Structure. In: Wang., L, Chen, K., Ong., Y.S. (eds.): Lecture Notes in Computer Science: Advances in Neural Computation, Vol. 1, Springer-Verlag, Berlin Heidelburg (2005) 750-759
12. Mackall, D., Nelson, S., Schumann, J.: Verification and Validation of Neural Networks of Aerospace Applications. Technical Report CR-211409, NASA (2002)
13. Martinez, T., Schulten, K.: Topology Representing Networks. Neural Networks 7(3) (1994) 507-522
14. Norgaard, M., Ravn O., Poulsen, N., Hansen, L.K.: Neural Networks for Modeling and Control of Dynamic Systems. Springer Verlag (2002)
15. Rumelhart, McClelland, and the PDP Research Group: Parallel Distributed Processing. MIT Press (1986)
16. Rysdyk R.,Calise, A.: Fault Tolerant Flight Control via Adaptive Neural Network Augmentation. AIAA-98-4483 (1998) 1722-1728
17. Schumann, J., Gupta, P.: Monitoring the Performance of A Neuro-adaptive Controller. In: Fischer, R., Preuss, R., and von Toussaint, U.: Proc. 24th International Workshop on Bayesian Inference and Maximum Entropy Methods in Sciences and Engineering (MAXENT), AIP (2004) 289-296
18. Schumann, J., Gupta, P., Jacklin. S.: Toward Verification and Validation of Adaptive Aircraft Controllers. In: Proc. IEEE Aerospace Conference. IEEE Press (2005)