Software Qual J
DOI 10.1007/s11219-007-9017-4

Validating neural network-based online adaptive systems: a case study

Yan Liu · Bojan Cukic · Srikanth Gururajan

© Springer Science+Business Media, LLC 2007

Abstract Biologically inspired soft computing paradigms such as neural networks are popular learning models adopted in online adaptive systems for their ability to cope with the demands of a changing environment. However, continual changes induce uncertainty that limits the applicability of conventional validation techniques to assure the reliable performance of such systems. In this paper, we discuss a dynamic approach to validate the adaptive system component. Our approach consists of two run-time techniques: (1) a statistical learning tool that detects unforeseen data; and (2) a reliability measure of the neural network output after it accommodates the environmental changes. A case study on NASA F-15 flight control system demonstrates that our techniques effectively detect unusual events and provide validation inferences in a real-time manner.

Keywords Validation · Online adaptive system · Novelty detection · Support vector data description · Validity index

1 Introduction

Adaptive systems are those systems whose functionality evolves over time due to environmental changes. If learning and adaptation are allowed to occur after the control system

Y. Liu (✉)
Motorola Labs, Motorola Inc, Schaumburg, IL, USA
e-mail: yanliu@motorola.com

B. Cukic
Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV, USA
e-mail: cukic@csee.wvu.edu

S. Gururajan
Mechanical and Aerospace Engineering Department, West Virginia University, Morgantown, WV, USA
e-mail: srikanth@web.cemr.wvu.edu

is deployed, the system is called online adaptive system (Mili, Cukic, Liu, & Ben Ayed, 2003). The use of biologically inspired soft computing systems for online adaptation to recuperate against changing system environment has revolutionized the operation of real-time automation and control applications. Neural networks are one of the most popular learning paradigms employed in online adaptive systems. Because the learning algorithms behind these computational architectures are usually derived from error/risk minimization theories, the computations are complex and the learning process contains non-linearity.

Adaptive flight control is considered one of the most challenging real-time automation and control applications 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 situation (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 validation approach. While adaptive systems in general are considered inherently difficult to validate, system uncertainties coupled with other real-time constraints make existing traditional validation techniques virtually useless for online adaptive systems and creating a suitable validation technique a challenging task (Mackall, Nelson, & Schumann, 2002; Schumann & Nelson, 2002).

Different from traditional types of software, neural networks are often viewed as black box models. They 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 adaptive 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 evaluation of its adaptation performance in a realistic operational real-time environment. The evaluation should be performed to examine: (1) how fast the neural network responds to and adapts to the changes; and (2) how well it accommodates the changes.

Our previous research using formal methods on certain families of neural networks suggests that environmental changes (learning data) have a significant impact on their adaptive behavior (Mili et al. 2003). Some violent changes may cause unstable learning behavior. Consequently, the adaptive component will produce unreliable output. In order to assure the learning performance and further validate the output, the changes must be observed, diagnosed and well understood. Moreover, the impact of such changes on adaptation and prediction performance also needs to be analyzed and measured during system operation.

We propose a novel validation approach that relies on robust and operative dynamic techniques. We use these techniques to examine: (1) the learning data on which the online adaptive component is trained, and (2) the neural network predictions after the completion of (partial) training. First, in order to detect certain changes, a real-time novelty detection technique is implemented before the data are fed into the adaptive component. Then, after the adaptive component learns and accommodates changes, it recalls what it has learned

and generates predictions. We define an online reliability measure to be associated with each neural network prediction for validity check.

1.1 Paper overview

This paper presents a dynamic validation approach consisting of two different techniques. Section 2 provides the application context as well as an overview of related research. The proposed validation methods are described in Sects. 3 and 4. In Sect. 5, we present experimental results obtained from an Intelligent Flight Control System (IFCS) simulator. Section 6 summarizes the proposed methods and describes future work.

2 Background

We present our validation approach in the application context of the intelligent flight control system designed and developed at NASA and specifically configured for an F-15 aircraft.

2.1 The intelligent flight control system

The Intelligent Flight Control System was developed by NASA with the primary goal to ''flight evaluate control concepts that incorporate emerging soft computing algorithms to provide an extremely robust aircraft capable of handling multiple accident and/or an off-nominal flight scenario'' [Boyd et al. 2001; Jorgensen 1991]. The diagram in Fig. 1 shows the architectural overview of NASA's IFCS implementation using Online Learning Neural Network (OLNN). The control concept can be briefly described as follows. Notable discrepancies from the outputs of the the Baseline Neural Network and the Real-time Parameter Identification (PID), either due to a change in the aircraft dynamics (loss of control surface, aileron, stabilator) or due to sensor noise/failure, are accounted by the OLNN. When there is such a change, a deviation between the desired and the actual state becomes significant. The OLNN is trained during operation to generate derivative corrections to minimize this deviation.

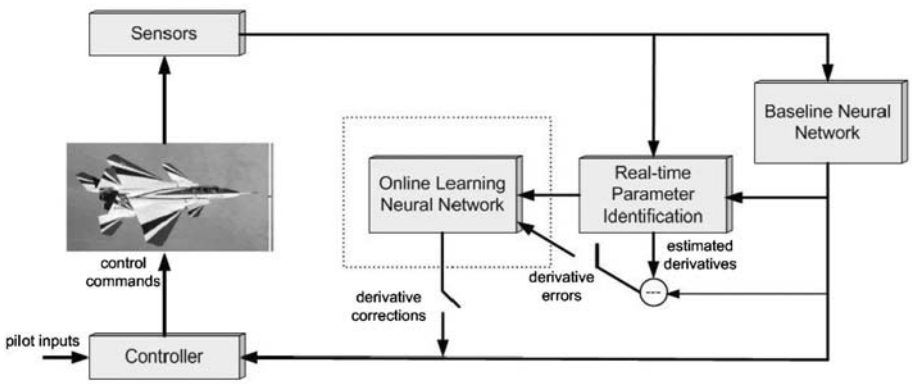

Fig. 1 The intelligent flight control system

The primary goal of OLNN is to accomplish in-flight accommodation of discrepancies, commonly known as Stability and Control Derivative errors. Such derivative errors indicate conditions that fall outside the scope of traditional (linearized) control gain look-up tables. When OLNN performs adaptation, its behavior has a direct consequence on the performance of the flight control system. As neural networks are often perceived as black-box computational tools, their learning behavior is highly complex and thus hard to validate. In such a safety-critical application, it is necessary to understand and predict the adaptive behavior of the OLNN. The critical role played by the online learning neural network in fine-tuning the control parameters and providing a smooth control adjustments is the motivation for the need for a practical, non-conventional validation methodology. Our goal of validating NN-based online component is to provide a means to detect novel (abnormal) conditions entering the OLNN, to investigate their impact on the NN's adaptive behavior, and to validate its predictions after the adaptation so that it ensures safe operation. In IFCS, the type of neural network that implements the OLNN is the Dynamic Cell Structure (DCS).

Our validation approach has been applied in the context of NASA IFCS program, which developed and flight evaluated described control concepts. IFCS program further investigates techniques to increase system robustness in terms of addressing accident and/or off-nominal flight scenarios.

2.2 Rleted work: validating online adaptive systems

While online adaptive systems are considered the most promising paradigm for improving quality of control applications, there is also a wide agreement that conventional V&V methodology is inapplicable to such systems. Because online learning systems may be used in life-critical (e.g. flight control) and mission-critical (e.g. power, aerospace) applications, they should be subject to strict quality standards, leaving a wide technological gap between the requirements of the application domain and the capabilities of available verification and validation technologies. A number of researchers dedicated their effort to proposing novel V&V methods to narrow this gap. A majority of published research focuses on methods that can be applied in an online fashion to assure the performance. Most proposed approaches adopt the online analysis/monitoring scheme to cope with the evolving performance of an online adaptive learner. These methods concentrate on three different aspects (phases) of online adaption.

1. For any learning system, training data is always gathered before the learner is used for prediction. Verification of the training data includes the analysis of its appropriateness and comprehensiveness. The strong emphasis on domain specific knowledge, its formal representation and mathematical analysis is suggested in (Del & Cukic, 2001). Del Gobbo and Cukic propose the analysis of the neural network with respect to conditions implying the existence of the solution (for function approximation) and the reachability of the solution from any possible initial state. Their third condition can be interpreted as condition for preservation of the learned information. This step is not fully applicable to on-line learning applications since training data are related to the real-time evolution of the system state, rather than the design choice. However, as proven by our previous investigation using formal methods (Mili et al., 2003), the training data has a very significant impact on system behavior. In a safety-critical system, the ability of ''novelty detection'' is crucial to system safety. It helps to detect suspicious learning data that is potentially hazardous to the system operation.

2. Online monitoring techniques have been proposed to validate the learning process. In a recent survey of methods for validating online learning neural networks, O. Raz (Raz, 2000) acknowledges that online monitoring techniques have a significant potential. Another promising research direction, according to Raz, is periodic rule extraction from an online neural network and partial (incremental) re-verification of these rules using symbolic model checking. In (Institute of Software Reseach, 2001), Taylor et al focus their effort on the Dynamic Cell Structure. They propose a prototype for real-time rule extraction in order to verify the correctness of DCS learning performance. In (Darrah, Taylor, & Skias, 2004), Darrah et al present rule extraction from DCS network learning and suggest future examination of performance based on such rules. Practical hurdles associated with this approach include determining the frequency of rule extraction and impracticality of near real-time model checking of complex systems. Yerramalla et. al. developed a monitoring technique for the DCS neural network embedded in the IFCS (Yerramalla, Cukic, & Fuller, 2003a; Yerramalla, Fuller, & Cukic, 2003b) based on Lyapunov stability theory. The online monitors operate in parallel to the neural network with the goal of determining whether (or not), under given conditions, the neural network is convergent, meaning that all state transition trajectories converge to a stationary state. The online monitor is theoretically founded and supported by an investigation of mathematical stability proofs that can define the engagement (or disengagement) of the online monitor.
3. Few research results target the validation of prediction performance, where the system is in operation after learning for a certain period of time. Schumann uses a Bayesian approach on Sigma-Pi networks to monitor the prediction performance for an online adaptive system in a real-time manner (Schumann & Gupta, 2004). In some cases, neural networks are modified to provide support for test-based (or online) validation of prediction performance. For example, Leonard et al (Leonard, Krammer, & Ungar, 1992) suggested a new architecture called Validity Index Net. A Validity Index network is a derivative of Radial Basis Function (RBF) network with the additional ability to calculate confidence intervals for its predictions based on the probability density of the ''similar'' training data observed in the past.

In this paper, we propose two validation methods to examine: (1) the learning data which the online neural network uses for training, and (2) the accuraccy of neural network predictions following the training cycles. Our parallel research (Yerramalla et al., 2003a; Yerramalla, Liu, Fuller, Cukic, & Gururajan, 2004) aims at validating the online learning process by developing online stability monitors and have shown a successful realization of convergence tracking of adaptation error towards a stable (or unstable) and safe (or unsafe) state in the IFCS. These three methods cover three phases of an online adaptive system and complete the framework for validating the IFCS as a typical example of neural network-based online adaptive system.

3 Novelty detection

In general, novelty detection techniques require beforehand knowledge of both nominal and off-nominal flight domains. However, for the validation of NN in IFCS it is impossible to anticipate all possible failure situations. As a one-class classification tool, Support Vector Data Description (SVDD) technique is derived from Support Vector learning theory (Vapnik, 1998) by Tax et al. (Tax & Duin, 1999b; Tax & Duin, 1999a). Differing from

general support vector classifiers that decide the maximum margin hyperplane to separate two classes, SVDD method tries to find an optimal decision boundary for a given data set. Thus, it provides the best possible representation of the target-class and offers inferences that can be used to detect the outliers from the nominal feature space. This, for our validation purposes, can be defined as the *''safe region''*.

SVDD is developed from the concept of finding a sphere with the minimal volume that contains all data items (Tax & Duin, 1999b). Given a data set S consisting of N examples x_i, $i = 1,.., N$, the SVDD's task is to minimize an error function containing the volume of this sphere. With the constraint that all data points must be within the sphere, which is defined by its radius R and its center a, the objective function can be translated into the following form by applying Lagrangian multipliers,

$$L(R, a, \alpha_i) = R^2 - \sum_i \alpha_i \{R^2 - (x_i^2 - 2ax_i + a^2)\}$$

where $\alpha_i > 0$ is the Lagrange multiplier. L is to be minimized with respect to R and a and maximized with respect to α_i.

In the solution that maximizes L, a large portion of α_i's become zero. The rest of α_i's are greater than zero and their corresponding objects are those called support objects. They lie on the boundary that forms a sphere that contains the data. Hence, object z is accepted by the description when:

$$|z - a|^2 = (z - \sum_i \alpha_i x_i)(z - \sum_i \alpha_i x_i) \leq R^2.$$

Flight control systems produce high dimensional data, characterized by non-linearity and, consequently, inseparability by a linear discriminant. This makes the data description more difficult to obtain. Similar to the Support Vector Machine (SVM) (Vapnik, 1998), by employing a kernel function, we are able to map the data from a high dimensional space onto a Hilbert space, also referred to as the ''feature space'', so data classification is achieved with reduced computational complexity.

After normalizing the learning data, we select the well-known Gaussian kernel function, i.e., $K(x,y) = exp(-|x-y|^2 / s^2)$, we now have:

$$L = 1 - \sum_i \alpha_i^2 - \sum_{i \neq j} \alpha_i \alpha_j K(x_i, x_j).$$

The formula of checking object z now becomes:

$$1 - 2\sum_i \alpha_i K(z, x_i) + \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) \leq R^2.$$

By applying the SVDD method, we are able to obtain a sound representation of the target class. To detect novelties (in our case, system failure conditions), a precise criterion should be inferred from empirical testing or pre-defined thresholds. Our previous research has demonstrated that SVDD can be adopted as an effective tool for novelty detection (Liu, Cukic, Menzies, Gururajan, & Napolitano, 2003) . Since the evaluation of test data points only involves ''support vectors'', a relatively small fraction of the data set, the detection of novelties becomes computationally efficient.

Furthermore, SVDD can also produces a ''posterior probability-like'' novelty measure for each testing data point that falls outside the boundary (Tax, 2001). Based on the assumption that the outliers are distributed uniformly in the feature space, Tax maps the distance from the outlier object to the defined decision boundary to a novelty measure. It is a quantified measure that indicates the degree of novelty of this particular object with respect to the target class. The mathematical definition of this mapping follows.

$$p(z|O) = exp(-d(z|T)/s)$$

where $p(z|O)$ is the probability that z belongs to the outlier class O; $d(z|T)$ is the distance from object z to the decision boundary obtained on the target class T in the feature space and s is the kernel width. SVDD also provides a flexible definition of ''novelty'' as we adjust the trade-off between sensitivity (the ability of determining true positives) and specificity (the ability of determining true negatives). Figure 2 illustrates an example of different boundaries provided by SVDD. The greater the distance from the innermost region, the rougher the boundary. Therefore, the sensitivity of outlier detection may be changed. In practice, a pre-defined threshold can be used as the furthest distance of a data point from the center, which the system can tolerate. Such pre-defined thresholds need sufficient testing within each specific data domain.

4 Validity Index in dynamic cell structures

Within an online adaptive system, the online neural network is in recall after a certain period of time it has been exposed to learning. However, it is possible that insufficient learning and/or data sparsity might cause locally poor fitting in certain regions. Within these regions, the online neural network is very likely to produce predictions with low

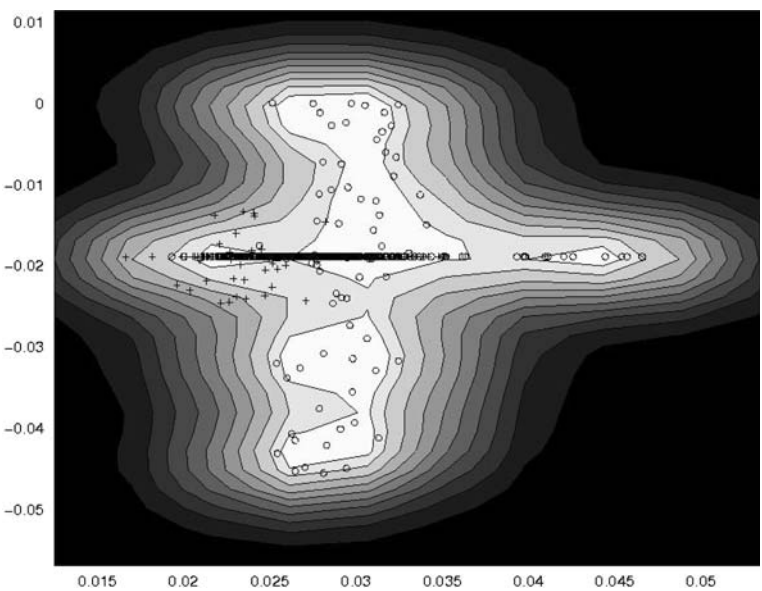

Fig. 2 SVDD with different distances from the center

confidence. In order to better evaluate its accommodation performance, we define a confidence measure to validate each output, namely, the validity index. Our method is inspired by J. Leonard's work in validating Radial Basis Function (RBF) neural networks (Leonard et al., 1992). He defines a reliability-like measure as the validity index for each output based on statistical analysis. The validity index is used for performance evaluation of neural network predictions.

4.1 Dynamic cell structures

The Dynamic Cell Structure (DCS) network can be seen as a special case of Self-Organizing Map (SOM) structures. The SOM has been introduced by Kohonen (Kohonen, 1990) and further improved to offer topology-preserving adaptive learning capabilities that can, in theory, respond and learn to abstract from a much wider variety of complex data-manifolds. 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 space. The input space is divided into different regions, referred to as the Voronoi regions (Bruske & Sommer, 1995; Martinetz & Schulten, 1994). As shown in Fig. 3,each Voronoi region is represented by its centroid, a neuron associated with its reference vector which is 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. Euclidean distance metric is used for finding BMUs and SBUs during the adaptation. The set of neurons connected to the BMU are considered its neighbors and denoted by NBR.

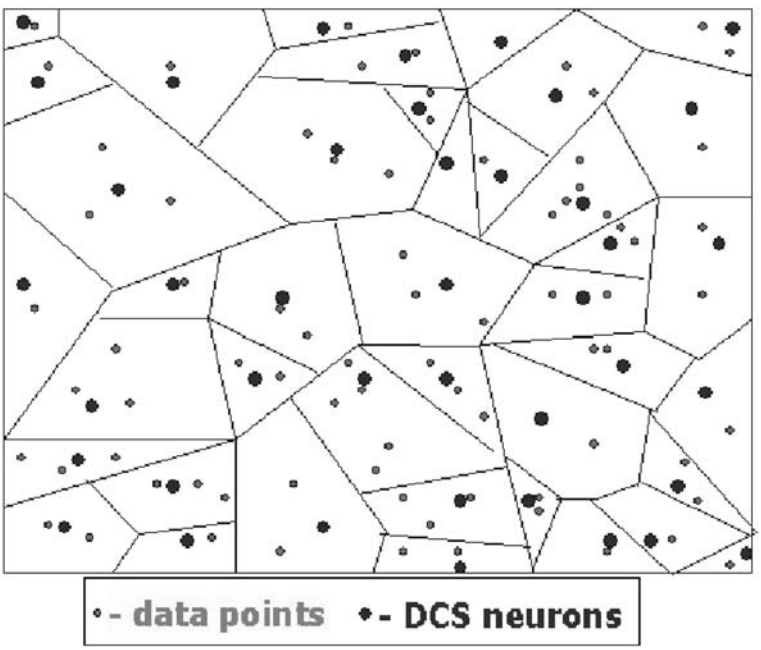

Fig. 3 Voronoi regions of a DCS network

In the DCS, the weight vectors of neurons and the lateral connections between neurons are updated based on the presented input data pattern. In addition, DCS adds neurons as needed to model the given data as accurately as needed. The resulting DCS network adopts the self-organizing structure that dynamically evolves with the presented data. The final structure formed by a DCS network after learning is topologically equivalent to the data set in the sense that during training the structure is adapting to the data and converging to a network that has a neighborhood preserving correspondence with the Voronoi regions of the data set. The Kohonen rule

$$\Delta w = \epsilon||m - w_{BMU(m)}|| \tag{1}$$

is used for updating the weight vectors w, and the Hebbian rule

$$C_{ij}(t+1) = \begin{cases} 1 & (i = BMU) \wedge (j = SBU) \\ 0 & (i = BMU) \wedge (j \in NBR - SBU) \wedge (C_{ij} < \theta) \\ \alpha C_{ij}(t) & (i = BMU) \wedge (j \in NBR - SBU) \wedge (C_{ij} \geq \theta) \\ C_{ij}(t) & i, j \neq BMU \end{cases} \tag{2}$$

is used for updating lateral connection values C_{ij} between neurons, where $0 \leq C_{ij} \leq 1$. Complete implementation details can be found in (Bruske & Sommer, 1995).

Different from the pre-defined static RBF network structure, the DCS progressively adjusts(grow/prune) its structure including locations of neurons and connections between them to adapt to the current learning data. Thus, unbiased estimation of confidence interval is impossible to be obtained through S-fold cross-validation due to constraint of time and space. Yet, DCS is more of a topological representation of the data than RBF. By the end of DCS learning, the data domain is divided into different Voronoi regions, of which every region has a neuron as its centroid. The ''locality'' of DCS learning is such that the output is determined by only two particular neurons, the best matching unit and the second best matching unit. Intuitively, if the Voronoi region of a neuron does not contain sufficient data, it is expected that the accuracy in that region will be poor. Based on the ''local error'' computed for each neuron, our approach provides an estimated confidence interval as the Validity Index for DCS output. The degree of novelty is reflected by the validity index and can be further verified through validity checks.

The DCS learning algorithm is depicted in Fig. 4. N is the number of training examples.Resource values are computed at each epoch as local error measurements associated with each neuron. They are used to determine the sum of squared error of the whole network. Starting initially from two connected neurons randomly selected from the training set, the DCS learning continues adjusting its topologically representative structure until the stopping criterion is met. The adaptation of lateral connections and weights of neurons are updated by Hebbian and Kohonen learning rules, respectively. The resource values of the neurons are updated using the quantization vector. In the final step of an iteration, the local error is reduced by inserting new neuron(s) into particular area(s) of the input space. The whole neural network is constructed dynamically. At the end of each learning epoch, the insertion or pruning of neurons is triggered, if necessary.

It should be noted that while the DCS network is in recall operational mode, the computation of output is different from that during training mode. In recall, given an input vector, the output is computed based on two neurons. One is the BMU of the input, the other is the closest neighbor of the BMU other than the SBU of the same input. In the

```
Initialization;

Repeat until stopping criterion is satisfied;
{
    Repeat N times

    {
        Determine the BMU and SBU;

        Update lateral connections;

        Adjust the weights;

        Update resource values;

    }

    If needed, a new neuron is inserted;

    Decrement resource values;
}
```

Fig. 4 A brief description of the DCS learning algorithm

absence of neighboring neurons of the BMU, the output value is calculated using the BMU only.

4.2 Computing validity index

We define the validity index (VI) in DCS networks as an estimated confidence interval of a DCS output, given the test input. The VI can be used to model the accuracy of the DCS network fitting and thus provide novelty 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 for to calculate the validity index in DCS. The computation of a validity index for a given input x consists of two steps: (1) compute the local error associated with each neuron, and (2) estimate the standard error of the DCS output for x using information obtained from step (1). The detailed description of these two steps follows.

1. The final form of DCS network structure is represented by neurons as centroids of Voronoi regions. Since the selection of the best matching unit must be unique, only those data points whose BMU are the same will be contained in the same region. Therefore, all Voronoi regions are non-overlapping and cover the entire learned domain. The data points inside each region significantly affect the local fitting accuracy. The local estimate of variance of the network residual in a particular region can be calculated over the data points contained in the region and then be associated with its representative neuron. The local estimate of variance s_i^2 associated with neuron i is computed as:

$$s_i^2 = \frac{1}{(n_i - 1)} \sum_{k=1}^{n_i} E_k,$$

where n_i is the number of data points covered by neuron i and E_k is the residual returned from the DCS recall function for data point k.Since the adjustment by competitive Hebbian learning rule concerns connections only between the BMU and its neighbors, the further update of weight values by Kohonen learning rule is performed only on the BMU and its neighbors. Consequently, training data points covered by the neighboring neurons of neuron i make proportional contributions to the local error of neuron i. Considering such contributions, we modify the computation of the local estimate of variance, now denoted by s'^2_i, as follows.

$$s'^2_i = \frac{s_i^2 + \sum_{j \in NBR} C_{ij} s_j^2}{1 + \sum_{j \in NBR} C_{ij}}.$$

As a result, the influence of all related data points is taken into account based on connections, referred to as C_{ij}, between the BMU and its neighbors.It should be noted that since the DCS networks within IFCS are trained online, no cross-validation is allowed. Hence, the residual calculated for each data point is in fact a biased estimate of the expected value of the residual due to the fact that each data point itself contributes to its own prediction. Nonetheless, under the assumption that there is no severe multi-collinearity and relatively few outliers exist in the data, the probability that the deviation from the expected value will be significant is very low and thus can be ignored.

2. Recall that the output produced by DCS is determined by the BMU and its closest neighbor (CNB) of the given input. Thus, the local errors associated with these two neurons are the source of fitting inaccuracy. As an estimate of this inaccuracy, we use the standard error, a statistic often used to place a confidence interval. Provided with the local estimate of variance for every neuron from step (1), we now define the 95% confidence limit for the local prediction error estimate with respect to neuron i as:

$$CL_i = t_{.95} \sqrt{1 + \frac{1}{n_i}} s'_i,$$

where $t_{.95}$ is the critical value of the Student's t-distribution with $n_i - 1$ degrees of freedom. The 95% confidence interval for the network output y, given a test input, is given by:

$$(y - \frac{(CL_i + CL_j)}{2}, y + \frac{(CL_i + CL_j)}{2}),$$

where $i = BMU$ and $j = CNB$ with respect to input x.

Finally, we slightly modified the DCS training algorithm in order 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, the validity index is computed based on the local errors and then associated with every DCS output. In the case of our application of interest, the IFCS, a domain specific threshold can be pre-defined to help verify that the accuracy indicated by the validity index is acceptable in the system context. This system performance validation step is enabled by the existence of the validity index.

5 Experimental results

Proper experimentation is required to justify realism and applicability of the proposed techniques in practice. With the aide of a high-fidelity flight simulator, we were able to test our techniques through extensive experimentation in simulated environments. The IFCS F-15 simulator has been developed by the research team at West Virginia University (Napolitano, Molinaro, Innocenti, & Martinelli, 1998). The control framework of the simulator is based on the IFCS architecture (See Fig. 1). Figure 5 shows a snapshot of the interface of the simulator. Through the simulator, we are able to collect valuable data representing nominal flight conditions as well as some failure scenarios.

5.1 Flight-data description

The simulation data depicts nominal and off-nominal flight conditions at a simulation rate of 20 Hz. A data frame is a point in a seven-dimensional space corresponding to 4 sensor readings (independent variables) and 3 stability and control derivative errors from PID and Baseline Neural Network (dependant variables). The NN tested here is the $DCS{-}C_z$ network, one of the five DCS-subnetworks of the IFCS. The independent variables are Mach number (the ratio of the speed of the aircraft to the local speed of sound), α (aircraft's angle of attack), β (side slip angle of the aircraft) and the altitude of the aircraft. The dependent variables are three stability and control derivative errors generated by the difference between PID and Baseline Neural Network.

In the following subsection, we present novelty detection using SVDD on the NN training data, which are normalized for better learning performance. The results of validity index in DCS network are described next. Both tools are tested on a particular failure mode

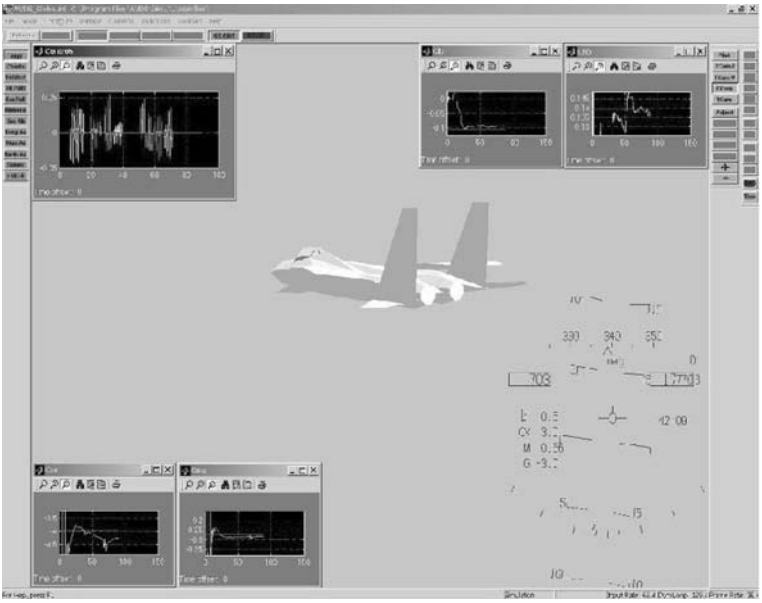

Fig. 5 NASA-WVF F-15 Simulator

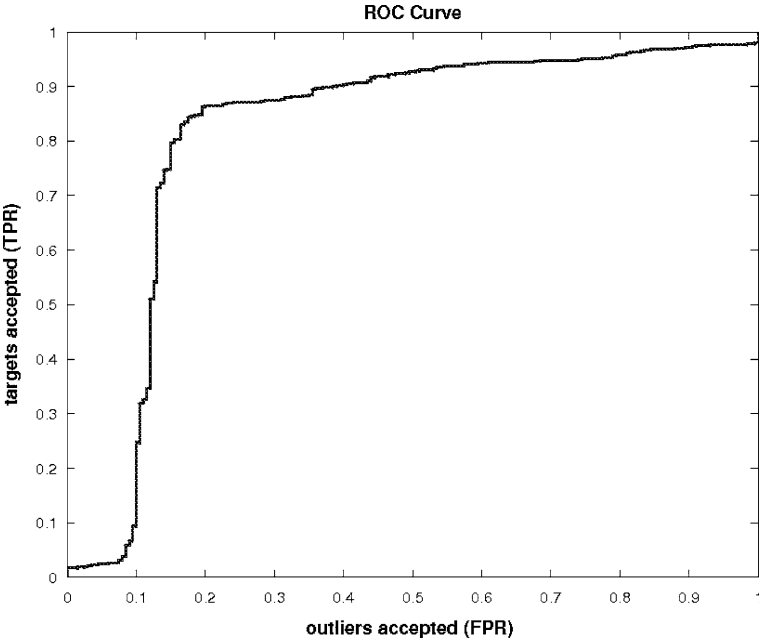

Fig. 6 The ROC curve of the SVDD tool

data set obtained from the simulator. A control surface failure (locked left stabilator, stuck at 0 degree) is induced at the 100th data frame.

5.2 Novelty detection using SVDD

We first simulate one run of nominal flight conditions of 40 seconds with a segment of 800 data points saved. After running SVDD on the nominal data, we obtain a sound data description of nominal flight conditions. A representative ROC curve is given in Fig. 6. By varying the value of the classification threshold we can obtain the differing SVDD classification characteristics in terms of combining false negatives and false positives. Based on the ROC curve and in line with system requirements, the specific operating point we choose for our SVDD tool is to allow 15% of nominal data classified as outliers.

We then use the boundary formed by SVDD to test the failure mode simulation. Novelty detection results are shown in Fig. 7b. Circles in Fig. 7a represent failure mode simulation data. The locked control surface failure results in input data points data falling outside the SVDD boundary. The novelty measures shown in Fig. 7b are probability-like measures computed for each data point based on the distance from the SVDD boundary. In this plot, *x*-axis represents the time frames and *y*-axis represents the novelty measures calculated by the SVDD tool. We can see from the plot that, after the 100th data frame, when the failure occurs, SVDD detects the abnormal changes and returns high novelty measures. This demonstrates effective and accurate detection capabilities of our SVDD detector.

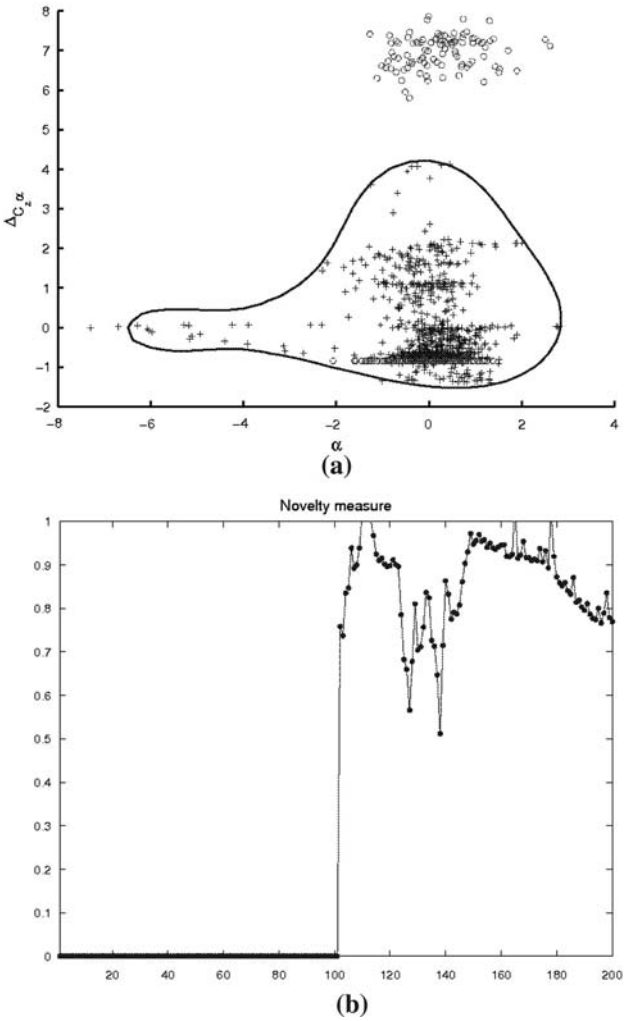

Fig. 7 Novelty detection. (**a**): SVDD of nominal flight simulation data is used to detect novelties. (**b**): Novelty measures returned by SVDD tool for each testing data point

5.3 Online testing of validity index

As part of the experimentation, we simulate the online learning of the DCS network under the failure mode condition. Running at 20 Hz, the DCS network updates its learning data buffer 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. While updating the data buffer, we discard the first incoming 20 data points and add the freshly generated 20 data points to maintain the buffer size, i.e., 200. The DCS network continues learning and repeats the recall-learn procedure.

Figure 8 shows the experimental results of our simulation on the failure mode condition. Plot (a) shows the final form of the DCS network structure at the end of the simulation. As a three-dimensional demonstration, the x-axis and y-axis represent two independent variables, α and β, respectively. The z-axis represents one derivative correction, $\Delta Cz\alpha$. 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. Plot (b) presents the validity index, shown as

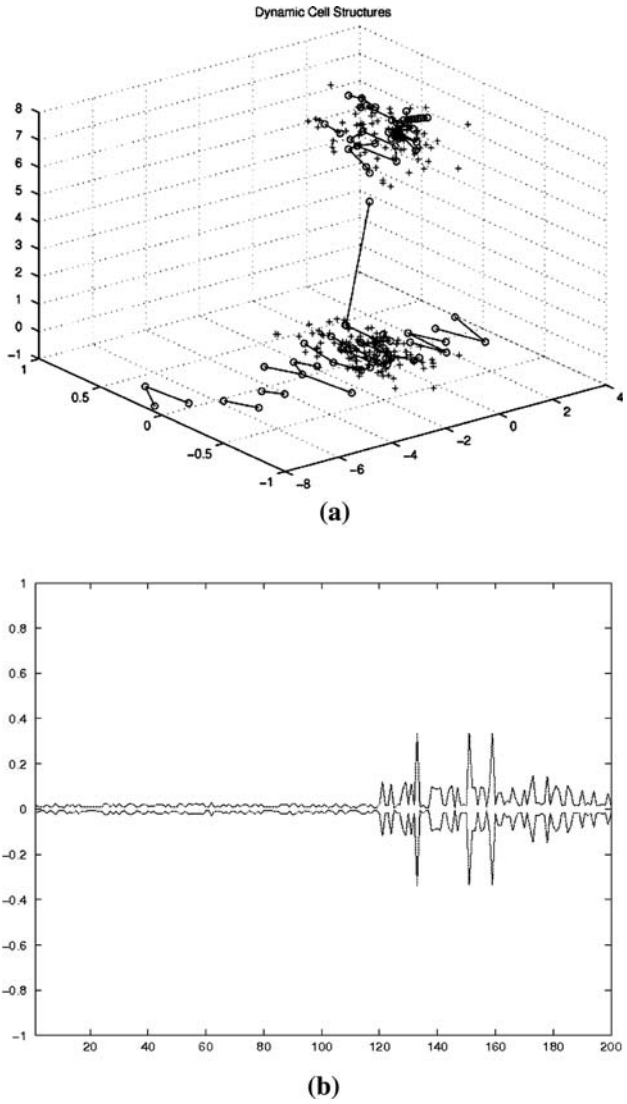

Fig. 8 Testing on failure mode simulation data in real-time (running at 20 Hz, failure occurs at 100th data frame). (**a**): The final form of DCS network structures. (**b**): Validity Index shown as error bars for each DCS output

error bars. The x-axis here represents the time frames. The failure occurs at the 100th data frame. We compute the validity index for the data points that are generated five seconds before and five seconds after the failure occurs. Plot (b) illustrates the validity index for 200 data points.

In our simulations, a trend revealed by the validity index indicates that increasingly larger error bars reflect failure occurrances. After the initial failure occurrance, the error bars shrink while the DCS network starts to adapt to the new domain, thus accommodating the failure. Thereafter, the change (increase/decrease) of the validity index varies. This variation depends 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 system failures.

6 Summary and conclusions

In this paper we propose a non-conventional approach for validating neural network based online adaptive systems. As software engineering researchers, the question we most frequently entertain is the following one: ''How does this work relate to software quality?'' In our research, indeed, we paid very limited attention to traditional software quality measures, such as process and product metrics, requirements, design or code analysis, etc. However, we did not overlook the good practices of software engineering. In the flight control domain, following rigorous software engineering lifecycle practices and traditional software quality assurance methods is mandatory. In our project, traditional quality assurance was not considered a problem. While trying to validate the ''quality'' of software that changes following its deployment, we had to develop completely new validation techniques. Our validation considers adaptive software as an integral part of the system. Any software quality assurance technique for adaptive systems must be tightly coupled in the context of system assurance. In essence, this type of assurance validates that the adaptive algorithms are sufficiently robust to cope with complex system environments. In mature technologies, trying out an algorithm while in doubt whether it can meet system performance objectives would represent a poor engineering practice. However, in emerging application domains, bold experimentation is necessary. Such experiments require significant departure from the traditional framework of software quality assurance too.

The proposed approach to validation of adaptive systems consists of two techniques that operate online. The first technique checks the novelty of learning inputs, while the second evaluates the validity of control outputs. Experimental results suggest that our approach provides the basis for the effective validation of the IFCS as a typical example of neural network-based online adaptive systems. Meanwhile, we observe a positive relationship between the novelty measures provided by the SVDD tool and the validity index in the DCS network output. We believe both tools can serve as online monitoring tools and provide validation inferences to further understand the adaptive behavior of other adaptive applications. Computational efficiency and scalability of both methods inspire our confidence that the proposed V&V method can be generalized to other type of neural networks. In fact, this point has been demonstrated in our recent publications (Yerramalla, 2005). In the future, we envision the application and further development of our techniques to many other adaptive applications.

References

Boyd, M. A., Schumann, J., Brat, G., Giannakopoulou, D., Cukic, B.,& Mili, A. (2001). Validation and verification process guide for software and neural nets, Technical report, NASA Ames Research Center, September.

Bruske, J. & Sommer, G. (1995). Dynamic cell structure learns perfectly topology preserving map. *Neural Computation, 7*(4) 845–865.

Darrah, M., Taylor, B., & Skias, S. (2004). Rule extraction from dynamic CEll structure neural networks used in a safety critical application. In *Proc. of the Seventeenth International Conference of the Florida Artificial Intelligence Research Society*, Miami Beach, FL, USA, September 2004.

Del Gobbo, D., & Cukic, B. (2001). Validating on-line neural networks. Technical report, Lane department of computer science and electrical engineering, West Virginia University, December 2001.

Institute of Software Reseach. (2001). Dynamic cell structure neural network report for the intelligent flight control system, Technical report, Document ID: IFC-DCSR-D002-UNCLASS-010401, January 2001.

Jorgensen, C. C. (1991). Feedback linearized aircraft control using dynamic cell structures. In*World Automation Congress* (ISSCI) (pp. 050.1-050.6). Alaska.

Kohonen T. (1990). The self-organizing map. *Proceedings. of the IEEE, 78*(9), 1464–1480.

Leonard, J. A., Kramer, M. A., & Ungar, L. H. (1992). Using radial bais functions to approximate a function and its error bounds. *IEEE Transactions on Neural Networks, 3*(4), 624–627.

Liu, Y., Cukic, B., Menzies, T., Gururajan, S., Napolitano, M. (2003). ''Validating an on-line adaptive system using support vector data description''. In *Proceedings of fifteenth international conference on tools with artificial intelligence*, Sacramento, CA.

Mackall, D., Nelson, S., & Schumann, J. (2002). Verification and validation of neural networks of aerospace applications, Technical report, CR-211409, NASA.

Martinetz, T., & Schulten, K. (1994). Topology representing networks. *Neural Networks, 7*(3), 507–522.

Mili, A., Cukic, B., Liu, Y., & Ben Ayed, R. (2003). Towards the verification and validation of on-line learning adaptive systems. In *Computational methods in software engineering*. Kluwer Scientific Publishing.

Napolitano, M., Molinaro, G., Innocenti, M., & Martinelli, D. (1998). A complete hardware package for a fault tolerant flight control system using online learning neural networks. Appears in *Proc. of the 1999 American Control Conference* (Vol. 4, pp. 2615–2619). San Diego, CA, USA, 1999.

Raz, O. (2000). Validation of online artificial neural networks–an informal classification of related approaches. Technical report, NASA Ames Research Center, Moffet Field, CA.

Schumann, J., & Nelson, S. (2002). Towards V&V of neural network based controllers. *Workshop on Self-Healing Systems*.

Schumann, J., & Gupta, P. Monitoring the Performance of a neuro-adaptive Controller. In *Proc. of the twentyfourth international workshop on Bayesian inference and maximum entropy methods in Sscience and engineering*, Garching bei München, Germany, 2004.

Tax, D. M. J., & Duin, R. P. W. (1999a). Data domain description using support vectors. In *Proc. european symposium on artificial neural networks, Bruges, April 21–23, 1999* (pp. 251–257). Brussels: D-Facto.

Tax, D. M. J., & Duin, R. P. W. (1999b). Support vector domain description. *Pattern Recognition Letters, 20*(11–13), 1191–1199.

Tax, D. M. J. (2001). ''One-class classification,'' *Dissertation*, ISBN: 90-75691-05-x.

Vapnik, V. N. (1998). *Statistical learning theory*. NY: Wiley.

Yerramalla, S., Cukic, B., & Fuller, E. (2003a). Lyapunov stability analysis of quantization error for DCS neural networks. *International joint conference on neural networks (IJCNN'03)*, Oregon.

Yerramalla, S., Fuller, E., & Cukic, B. (2003b). *Lyapunov analysis of neural network stability in an adaptive flight control system. Sixth symposium on self stabilizing systems (SSS-03)*. San Francisco, CA, June 2003.

Yerramalla, S., Liu, Y., Fuller, E., Cukic B., & Gururajan, S. (2004). An approach to V&V of embedded adaptive systems. In *Lecture notes in computer science (LNCS) Proceeding of third NASA-Goddard/ IEEE workshop on formal approaches to agent-based systems*, Springer-Verlag.

Yerramalla, S. (2005). Tability monitoring and analysis of online learning neural networks. Doctoral dissertation. WV: West Virginia University.

Author Biographies

Yan Liu received the BS degree in computer science from Wuhan University, China, and the MS and PhD degrees in computer science from West Virginia University. She is currently a research scientist at Motorola Labs, Motorola Inc. Her research interests are in the areas of software V&V, machine learning and statistical learning.

Bojan Cukic received his PhD in Computer Science from the University of Houston.Currently, he is a Robert C. Byrd associate professor in the Lane Department of Computer Science and Electrical Engineering at West Virginia University, where he also serves as a co-director of the Center for Identification Technology Research, an NSF Industry University Cooperative Research Center.His research interests include software engineering for high-assurance systems, fault-tolerant computing, information assurance, and biometrics. He received a US National Science Foundation Career award and a Tycho Brahe Award for research excellence from NASA Office of Safety and Mission Assurance. Dr. Cukic is a member of the editorial board of Empirical Software Engineering and a member of the steering committee for the IEEE International Symposium on Software Reliability Engineering (ISSRE).

Srikanth Gururajan received his Bachelor's degree from the University of Madras in 1997 and his Master's degree in Mechanical Engineering and Doctoral degree in Aerospace Engineering from West Virginia University, Morgantown, West Virginia in 1999 and 2006. His interests are in the areas of design, building and flight testing of Unmanned Aerial Vehicles, flight control systems, neural networks and high performance computational clusters.