(Preprint) AAS 19-703

PREDICTING SATELLITE CLOSE APPROACHES USING STATISTICAL PARAMETERS IN THE CONTEXT OF ARTIFICIAL INTELLIGENCE

A. Mashiku*, C. Frueh[ŧ], N. Memarsadeghi[¥], E. Gizzi[‡], M. Zielinski[£] and A. Burton[£]

In order to ensure a sustainable use of low earth orbit in particular and near Earth space in general, reliable and effective close approach prediction between space objects is key. Only this allows for efficient and timely collision avoidance. Space Situational Awareness (SSA) for commercial and government missions will be facing the rapidly growing amount of small and potentially less agile satellites as well as debris in the near earth realm, such as the increase in CubeSat launches and upcoming large constellations. At the same time, space object detection capabilities are expected to increase significantly, allowing for the reliable detection of smaller objects, e.g. when the Air Force Space Fence radar becomes operational. In combination, the space object catalog is expected to increase tremendously in size. In this paper, we introduce an investigative approach based on the latest capabilities in artificial intelligence in fostering the potential for fast and accurate close approach predictions. We consider the study of statistical and information theory parameters in contrast and complementary to the classical probability of collision computation alone, in order to determine the feasibility of reliably predicting close approaches.

INTRODUCTION

As the number of objects in near-Earth space increases, so does the importance of developing techniques for the rapid and accurate assessment of space object conjunction events. Collision avoidance decisions currently hinge on the probability of collision (Pc) as a key component in risk assessment, but common methods for computing Pc make significant assumptions about the geometry of the encounter that may not always hold in certain situations[1,2,3]. Additionally, Pc is strongly influenced by the state uncertainty present in the system, which is time-varying as the space objects are propagated and observed[1]. The most accurate way for performing conjunction analysis between two spacecraft is through a Monte Carlo (MC) simulation of the state space of both the primary and secondary objects[2,3].

(*Aerospace Engineer, ¥Computer Engineer), NASA Goddard Space Flight Center, 8800 Greenbelt Rd, Greenbelt, MD 20771, alinda.k.mashiku@nasa.gov, nargess.memarsadeghi-1@nasa.gov

(ŧAssistant Professor, £Graduate Student,) School of Aeronautics and Astronautics, Purdue University, 701W. Stadium Ave. West Lafayette, IN 47907-2045, (cfrueh@purdue.edu, mzielin@purdue.edu , burton30@purdue.edu)

‡Graduate Student, College of Engineering, Tufts University, 419 Boston Ave, Medford MA 02155, Evana.Gizzi@Tufts.edu

However, even when employing parallel computing and simplified prediction models, MC simulations may not always be feasible for routine scans of the catalog for close approaches due to the long run times and the number of samples required. As a result, Pc computations are employed instead that use a projection of the uncertainty volume into relative space. Those computations can be performed reasonably fast, however the associated 2-dimensional assumptions are often not applicable to the realm of low relative velocity encounters, in which the objects do not pass through each other's uncertainty volume in a timely manner[27]. This situation may pose challenges for current methods of collision prediction with the expected increase in space objects.

In this paper, we introduce an investigative approach based on the latest capabilities in artificial intelligence, or more specifically machine learning, in fostering the potential for fast and improved close approach predictions. We consider the study of statistical and information theory parameters in contrast with and complementary to the classical Pc computation alone, in order to determine the feasibility of reliably predicting close approaches. We consider the development of a set of "information parameters" that would serve as a supplement to the Pc in the conjunction assessment process, a tool that can be used to examine the evolution of Pc and other parameters during the conjunction events, and a set of conjunction event data to be used for training machine learning algorithms. Preliminary work on reducing state uncertainty by scheduling observations so as to maximize information gain is also described and presented in this paper as an approach to capture the information content as a scalar value or an informational parameter.

The initial approach considered the construction of a neuro-fuzzy-logic-based decision making system that is based on intelligently selecting parameters beyond the Pc values to take the comprehensive knowledge of data information gain from measurement processing, orbit generation and object dynamics into account. Fuzzy logic is a form of decision-making logic that uses functions that produce partial truth membership values that range between the standard Boolean truth values of 1 and 0. These values are used to construct a Fuzzy Inference System (FIS), an approach to logic designed to mimic human intelligence. We compare the inference for an impending close approach with weighted assignments of the parameters using several models of the FIS. The values for these weights were incorporated using unsupervised-machine learning clustering techniques to aid in inferring a close approach conjunction analysis. We investigated using the information parameters in a machine learning construct to help assess the close conjunctions and in turn infer the collision avoidance decision-making process. Specifically, we investigated how a FIS could process those parameters to provide a conjunction assessment output, in either a standalone approach or in a way that could enhance the Pc-based decision construct.

FIS are able to capture partial memberships of variables into different sets, and generate outputs that exists in a continuous space, versus discrete classifications, which is characteristic of classical logical systems and of many traditional machine learning constructs. The hypothesis herein lies with the expected benefit in resolving the collision avoidance decision ambiguities that exist in the parameter set level that are within the close neighborhood of one another. A similar construct was investigated that introduced a single risk index, known as the F-value that aggregated all considered risk figures of merit[4]. The parameters considered for the F-value construct were risk and quality assessment parameters but leveraged the fuzzy-logic approach. In our approach, we implemented unsupervised machine learning clustering algorithms using K-means and Support Vector Machines (SVM) methods to determine whether the parameters correlated to ground truth classifications as well as use their performances to obtain the correct weights for the FIS.

We also investigated constructing a Deep Neural Network (DNN) model, and trained the model using parameter combination sets based on their performance from the unsupervised clustering methods. The performance of the DNN was assessed to determine the correct assessment of risk assignment of the test cases that were considered. The goal was to assess whether the resulting outputs could be understood, applied, and implemented as a conjunction assessment decision-making tool for close approaches. For the ground truth, MC simulations were used for both simulated and real cases from the NASA Conjunction Assessment Risk Analysis (CARA) program data used for this study.

APPROACH

Machine Learning algorithms require large, diverse sets of quality data in order to come to meaningful conclusions. The preliminary approach leveraged the use of 11 of the 12 baselined cases of simulated data from Alfano et al[3] (excluded case 9, due to its geometric similarity to case 10 apart from the integration duration as presented in Reference 3). These cases were considered due to their appeal with respect to their diverse orbital and close approach characteristics as well as how they affect the reliability of the 2D Pc in various paradigms[5]. The ground truth was established using MC samples generated for each case based on a Gaussian random sampling of the variances.

Additionally, we obtained around 450,000 cases of historical conjunction data from the CARA program. One of the main challenges of using historical data was the availability of a significant number of cases that could be categorized as close approaches. A significant portion of the historical data contained conjunctions that did not require any Risk Mitigation Maneuvers (RMM) and were thus deemed to be non-close approaches, or safe encounters, based on the Pc risk threshold.

For this preliminary approach, we selected 20,000 cases and applied various scaling factors that altered the Pc output in order to have a broader spectrum of resulting Pc outputs to work with to compare the categorization of safe and unsafe close approaches. A wide array of cases were generated that captured Low Earth Orbits (LEO), Geosynchronous Earth Orbits (GEO), Medium Earth Orbits (MEO), and High Earth Orbits (HEO). The distribution of these 20,000 cases were observed not only in the Pc values but also the resulting miss distances, relative velocities, approach angles, as well as the variability in the Primary and Secondary orbit eccentricities. For the 20,000 cases considered, we implemented a simulated measurement update using the Kalman Filter for the LEO and MEO orbits and the Unscented Kalman Filter for the GEO and MEO orbits to obtain 3-sets of information at the time of closest approach (TCA). The information of interest for our preliminary analysis that provides the bedrock for the information parameters are the state and covariance information at TCA. The goal was to have a varied set of data that contains simulated measurements updated at varied instances prior to TCA to observe how much the information content varies when propagated and investigate how sensor tasking knowledge can be used to infer the decision making process at TCA[6], as shown in Figure 1.

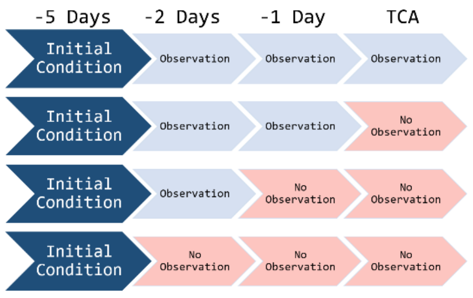


Figure 1: Schematic illustrating the observation and propagation scheme used in the forward propagation (measurement update) script

In order to evaluate the information gain compared to the knowledge of the state resulting from the simulated measurement updates, we leveraged the Kullback-Leibler information gain[7] method. The Kullback-Leibler information gain (G_{KL}) is related to the Kullback-Leibler divergence (KLD) and quantifies the information gain resulting from an observation by comparing the pre- and post-update covariance matrices. The formula for the measure is given in Equation (1), where $\boldsymbol{P}^-$ is the pre-update covariance matrix and $\boldsymbol{P}^+$ is the post-update covariance matrix. The Kullback-Leibler information gain is negative for an observation that increases the uncertainty of the state estimate and positive for an observation that decreases the uncertainty of the state estimate.

$$G_{KL} = \frac{1}{2}\log\frac{|\boldsymbol{P}^-|}{|\boldsymbol{P}^+|} \tag{1}$$

Figure 2, displays the G_{KL} values associated with a series of observations along with the largest eigenvalues of the post-update covariance matrices, to illustrate the relationship between positive information gain and shrinking uncertainty.

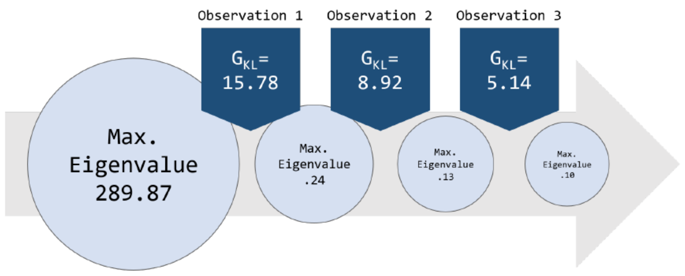


Figure 2: The relationship between positive information gain and shrinking uncertainty based on comparing the largest eigenvalue of post-update covariance matrices to the Kullback-Leibler information gain associated with the corresponding observations

These instances of simulated updated information, provided unique sets of information quality at TCA for information parameter set generation. Expectedly, those with the largest overall G_{KL} will provide more accurate information content at TCA compared to those with a smaller overall G_{KL}. The goal for this implementation was to determine how available sensor tasking information can ultimately be used as an input parameter when leveraging machine learning techniques in conjunction assessment of close approaches for decision making. Future work in this area will investigate the incorporation of sensor tasking in the overall machine learning paradigm.

INFORMATION PARAMETERS

The project set out to identify additional “information parameters” derived from the primary and secondary objects’ state estimates that may be able to provide insight into conjunction events that cannot be obtained from Pc alone. It is hoped that parameters might be identified that can help distinguish between different trends in Pc evolution or supplement Pc in situations in which the assumptions behind 2D Pc are not supported.

An initial set of six information parameters and their variants were considered here. These parameters were based on statistical measures that were used to describe relationships between probability distributions and their random vectors or state, as well as parameters that provide information about the geometry of the conjunction.

Mahalanobis distance (MHD)

The Mahalanobis distance can be interpreted as either the number of standard deviations between a point (or a state) and the mean of a probability distribution or the dissimilarity of two random vectors that are described by the same distribution[9]. Both interpretations are equivalent for the conjunction assessment scenario (distance of the relative state from the mean of the combined uncertainty distribution, dissimilarity of the two state vectors that are both described by the combined uncertainty distribution). The Mahalanobis distance was calculated as shown in Equation (2), where μ_p and μ_s are the state vectors of the primary and secondary objects, respectively, and $P_{comb}^{-1}$is the combined covariance matrix.

$$D_M = \sqrt{(\mu_p - \mu_s)^T P_{comb}^{-1} (\mu_p - \mu_s)} \tag{2}$$

Kullback-Leibler divergence (KLD)

Kullback-Leibler divergence is an asymmetric measure of the divergence between two probability distributions[7]. The standard form is shown in Equation (3), where p(x) and q(x) are probability density functions.

$$D_{KL}(P||Q) = \int p(x) \log \frac{p(x)}{q(x)} dx \tag{3}$$

A closed-form expression of the KLD for Gaussian distributions is shown in Equation (4), where P_p and P_s are the covariance matrices for the primary and secondary objects, respectively, and $\Delta\mu$ is the relative state of the two objects[8].

$$D_{KL} = \frac{1}{2}\left(\log \frac{|P_s|}{|P_p|} - n + trace\left(P_s^{-1} P_p\right) + \Delta\mu^T P_s^{-1} \Delta\mu\right) \tag{4}$$

Bhattacharyya distance (BD)

The Bhattacharyya distance is a symmetric measure of the divergence between two probability distributions[10]. The standard form is shown in Equation (5).

$$D_B = -\ln\left(\int \sqrt{p(x)q(x)}\, dx\right) \tag{5}$$

The Mahalanobis distance is a specific case of the Bhattacharyya distance that arises when the two distributions have the same standard deviation. The Mahalanobis distance is clearly visible in the first term of the closed-form expression of the Bhattacharyya distance for Gaussian distributions shown in Equation (6).

$$D_B = \frac{1}{8}\, \Delta\mu^T P_{comb}^{-1} \Delta\mu + \frac{1}{2} \ln\left(\frac{|P_{comb}|}{\sqrt{|P_s||P_p|}}\right) \tag{6}$$

L2 norm

Commonly known as the Euclidian norm, the L2 norm gives the distance between the two objects' state vectors[11]. If only the position portion of the state vectors are used to calculate the L2 norm it describes the geometric distance between the two objects' means, otherwise known as the miss distance. The L2 norm is calculated according to Equation (7), where n is the number of states being considered.

$$|\Delta\mu| = \sqrt{\sum_{k=1}^{n} |\Delta\mu_k|} \tag{7}$$

Orbit angle (OA)

The angle between orbital planes is calculated as the angle between the cross track portions of the objects' radial-in track and cross track (RIC) position vectors.

Miss distance (MD)

The miss distance is the Euclidean distance between the primary and secondary space objects. This is similar to the evaluation on Equation (7) when considering the positions of the state.

UNSUPERVISED MACHINE LEARNING: CLASSIFICATION AND CLUSTERING METHODS

Unsupervised learning methods classify data into groups based on features within the dataset that may not be immediately obvious. The data fed into unsupervised learning algorithms are not initially labeled with their classifications, but rather the algorithm imposes its own functionality onto the data to generate those classifications. The intention of this initial approach of using unsupervised machine learning on the information parameter set, was to explore the data and determine if an internal representation existed that would cluster the parameter sets into two categories: safe or close encounter(not safe).

There were two reasons for performing clustering methods on our data. The first was to see if the features used to separate the data from such methods would give way to natural separation that correlated with our ground truth. We did this by comparing the output of the processes with the discretized ground truth output set, to see how well the methods performed. That is, to see if there was promise in the ability to separate the data based on features related to unsafe close approaches as evidenced in the quantification of statistical information parameters. The second reason was to find naturally characterizing metrics on the data and use the performance metrics as weights to inform the actual construction of our Fuzzy Inference Systems.

Clustering algorithms fall into two broad groups[12]:

-Hard Clustering, where each data point belongs to only one cluster ex. K-means and Support Vector Machines

-Soft Clustering, where each data point belongs to more than one cluster ex. Fuzzy C-means and Gaussian Mixture Models

For CA applications, decision making mostly tends to be a binary-decision; to maneuver in order to mitigate a dangerous close encounter or not, with the understanding that the close encounter is deemed safe. This can be construed as a hard clustering approach, however other factors are always taken into consideration such as the missions capability to perform such a maneuver, the orbit determination quality, the sensor tasking applied on the secondary object and so on. Therefore, if we can augment the binary decision output by finding a way to incorporate additional information by intelligently weighing in the parameters based on their performance with respect to a Monte Carlo run Pc value, we could investigate the contribution and effectiveness of these parameters for decision making.

We implemented a hard clustering approach for this preliminary investigation and used the K-means clustering methods and the Support Vector Machines classification methods, described below.

K-means clustering

K-means clustering partitions data into clusters based on the closest mean of a chosen number of centroid points, which are representative of classes. For our K-means clustering, we chose 2 centroids ($K = 2$) for each parameter, representing a “close encounter/not safe” and a “safe” categorization. Execution of K-means clustering resulted in the following values:

1. K-means Centroid Values corresponding to the classes
2. K-means Standard Deviations around the centroids
3. K-means Performance Metric generated by calculating the percentage of the dataset that was correctly assigned to the ground truth (see Table 1)

Support Vector Machines (SVM) classification

SVM-classification techniques separates data by using kernel methods to extended its dimensionality in order to find separations in the data that are not existent in its distribution in lower dimensionality spaces. Execution of SVM classification methods resulted in the following values:

1. SVM Standard Deviations around the centroids
2. SVM Performance Metric generated by calculating the percentage of the dataset that was correctly assigned to the ground truth (see Table 1)

Table 1. Clustering Performance Methods for K-means and SVM using the Performance Metric

Parameter	k-means	SVM
Probability of Collision (Pc)	0.7742	0.9995
Miss Distance (MD)	0.6389	0.8314
Mahalanobis Distance (MHD)	0.6983	0.8810
Bhattacharyya Distance (BD)	0.7611	0.8864
Kullback-Leibler Distance (KLD)	0.7736	0.8459
Orbit Angle (OA)	0.5387	0.8711

The output data from the SVM-classification had an overall higher correlation with the discretized ground truth value set than the K-means clustering method as shown in Table 1. Because of this, we chose to use SVM standard deviation and performance values to inform the construction of our Fuzzy Inference System. This will be discussed in further detail in the next section of this paper.

Supervised learning methods use data with ground truth classifications to train a "classifier" construct. These methods require a "training" dataset of input-output pairs and will be demonstrated in our DNN models constructs.

FUZZY INFERENCE SYSTEMS CONTEXT

Fuzzy inference systems map input to output using fuzzy logic functions, which express partial membership of variables or parameters to certain sets. FIS's are used mostly in decision making problems, where there is not a concise certainty on whether an input belongs to a discretized set[13,14]. These systems can be trained or untrained, and manually constructed or constructed through learning on a dataset. In our work, we focused mostly on manually constructed systems which were trained after classification for model fitting purposes. Figure 3 below shows a snapshot of the graphical user interfaces (GUI) of the FIS using MATLAB's Fuzzy Logic Toolbox[15].

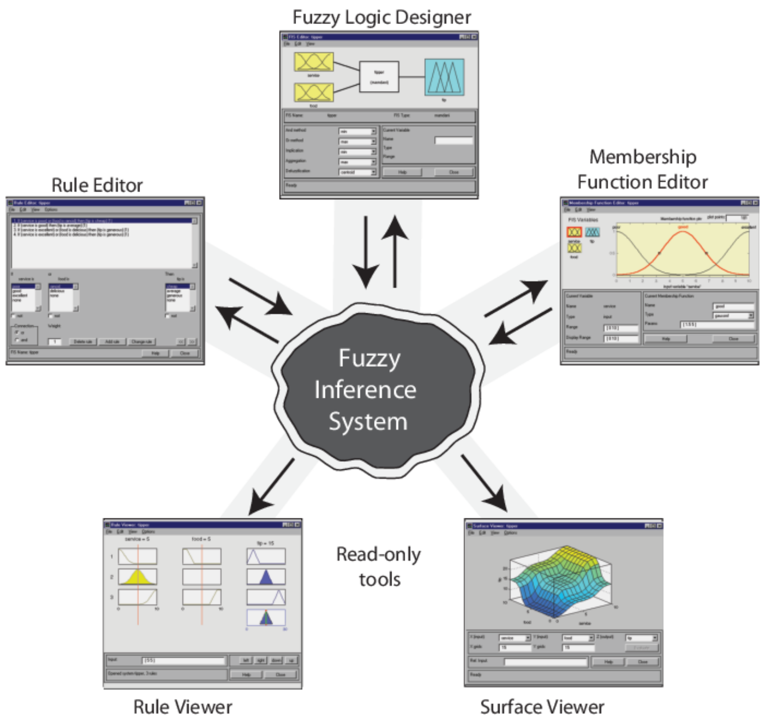


Figure 3. Fuzzy Inference System User Interface using MATLAB®

MATLAB Fuzzy Inference System's GUIs has tools that allow you to build, edit and view the FIS. The Fuzzy Logic Designer is where all the Fuzzy Membership Functions (FMF) are defined and each information parameter input can be defined as a FMF. The Rule editor is where the performance metrics from the SVM are used to construct the decision making process to produce an output decision. More details can be found in Reference 15.

There are three types of FIS: Mamdani FIS model, Sugeno type FIS model and the Adaptive Neuro-Fuzzy Inference Systems.

Mamdani FIS model[15]

Mamdani-Type (Mamdani) fuzzy inference systems are fully constructed by the user. These systems have a set of basic components, including a set of input variables and output variables. Each variable has a set of associated FMF that define a specific input's degree of membership to each of the set defined in terms of that variable. Mamdani systems also have a set of “if-then rules”, similar to classical, non-fuzzy logical systems. However, they differ in that the inputs of FIS’s are values between 0 and 1, whereas those of classical logical systems take on discretized Boolean values. Therefore in FIS’s, there can be partial truth to holding and defined rules. Once the output spaces of the rules are generated, these spaces are aggregated together and "de-fuzzified" with a method that can take the aggregated space, and flatten it into a single quantifying value output.

Sugeno FIS model

The Sugeno FIS model method is similar to the Mamdani method’s fuzzy inference process in the fuzzifying of the inputs and applying the fuzzy operator.[16,17] The main difference between Mamdani and Sugeno is that the Sugeno output membership functions are either linear or constant.[16,17]

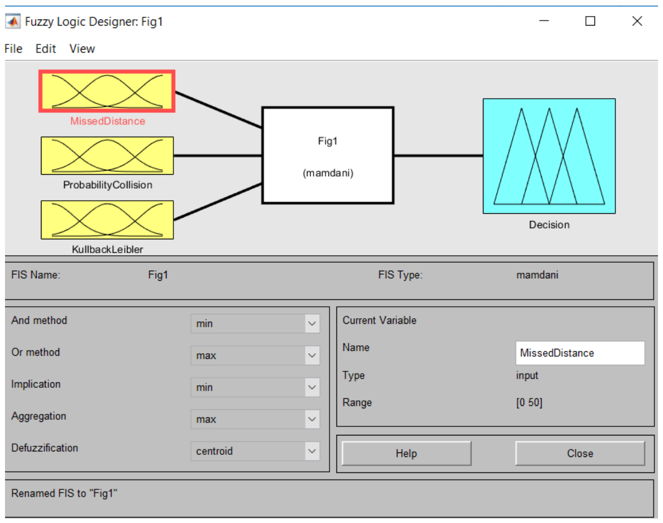
Adaptive Neuro-Fuzzy Inference System (ANFIS) Model

Adaptive Neuro-Fuzzy Inference Systems are very similar to Mamdani FIS’s, except that they do not rely on a predetermined structure. The mechanisms that underly this system use modeling techniques to infer structure in a way that is similar to the training that happens with neural networks. ANFIS systems will result in tweaked membership functions and parameters on those functions.

Sample Mamdani FIS model implementation

In constructing the Mamdani FIS prototype model, it was important and imperative that we use the informational parameters that provided us with a physics-based explanation in order to validate and verify the outputs of the FIS model.

We used the simulated data set by Alfano[3], and used three information parameters: miss distance, Pc, and Kullback-Leibler Divergence. We constructed the FMF for each parameter based on the clustering outputs of the values. For example, a small miss distance will infer a close encounter and thus we labeled the output as 0 for unsafe and a large miss distance will be the contrary with an output label of 1 for safe. A similar approach was implemented for both the Pc and the KLD.



FIS Name: Fig1
FIS Type: mamdani

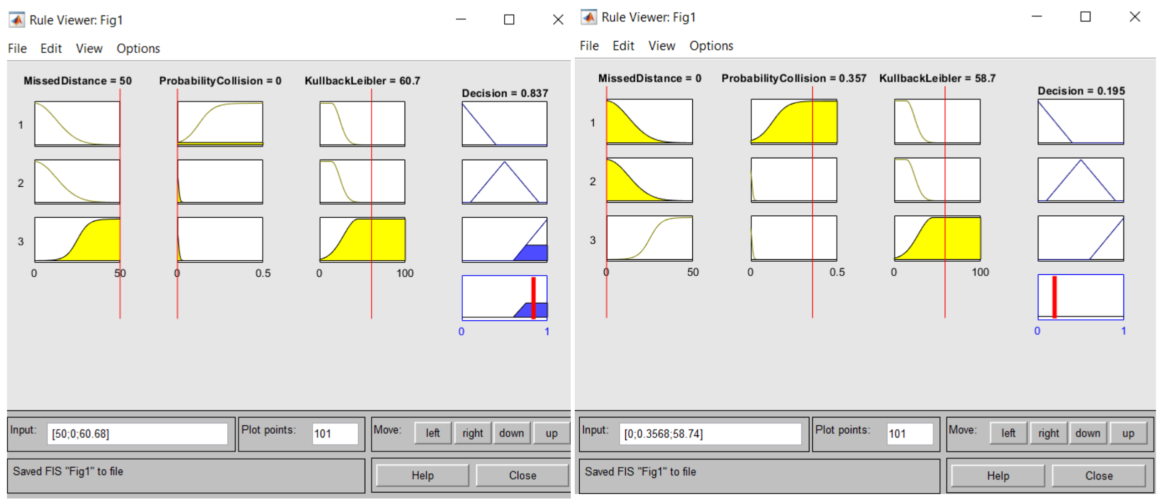
And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid

Current Variable
Name: MissedDistance
Type: input
Range: [0 50]

Renamed FIS to "Fig1"

Figure 4. Fuzzy Logic Designer GUI using MATLAB® defining the FMFs Missed Distance, Probability of Collision, and the Kullback-Leibler Divergence.

In Figure 4, the Mamdani Fuzzy Logic Designer’s rules are created to map the values of the parameters to the FMF and designate an output with respect to the weights from the unsupervised machine learning models. This provides a prototype for a multi-parameter based decision-making tool constructed using fuzzy-logic rules. The FIS Rule Viewer is shown in Figure 5, where each information parameter has a FMF whose information contributes to the multi-parameter decision output. The vertical red lines show the sliding values of the parameters and how they each affect the final decision of {0,1} = {unsafe, safe}. An interesting observation here is that the KLD parameter range value inputs did not affect the final decision. This was because the KLD parameter was a measure that compared the probability density function (PDF) or uncertainties of the primary object *to* the secondary object, not a specific measure that assesses a close approach situation. This shows that it is imperative that each contributing parameter considered must provide added information value. It was eventually clear that the inherent comparisons of the primary and secondary uncertainties did not directly provide information with respect to conjunction assessment rules despite a larger weight value from the SVM outputs.



Input: [50;0;60.68]

Saved FIS "Fig1" to file

Input: [0;0.3568;58.74]

Saved FIS "Fig1" to file

Figure 5. FIS Rule Viewer providing the decision output. Figure on the left with a large miss distance and low Pc provides a decision output of 0.837 (~ 1 for safe) Figure on the right with a small miss distance and high Pc provides a decision output of 0.195 (~0 for unsafe)

More investigation is needed to determine the correct combination of information parameters to construct a FIS decision making system. In our work, we discovered that not all combinations of informational parameters provided intuitive or expected outputs. This study will be revisited with a follow-on investigation on the applicability of considering the use of FIS for CA decision making.

DEEP-NEURAL NETWORK MODEL CONTEXT

We also took the approach to design and implement a Deep Neural Network (DNN) model for decision making augmentation with Pc. A DNN is based on Deep Learning, which is a type of machine learning that is usually implemented using a neural network architecture. The term "deep" refers to the number of hidden layers implemented in the network that can have from 2 or 3 to hundreds of hidden layers[19,20]. The DNN model is trained with a subset of our dataset that uses the known outputs given the input information parameters to construct the DNN model. One of the benefits of using a DNN is that it learns the features and classifiers automatically with unlimited accuracy[19]. However, this automatically implies the availability of large sets of data with good quality data.

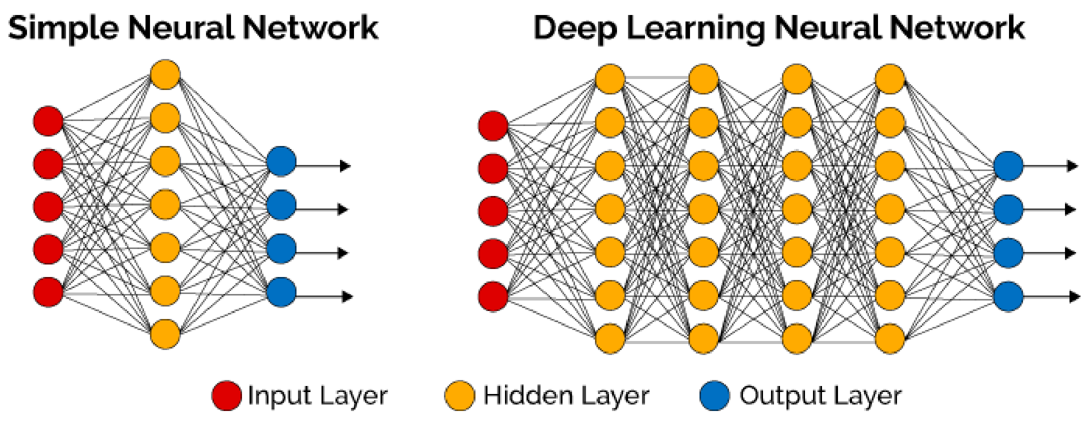


Figure 6. A simple neural network compared to a deep learning neural network based on the numbers of hidden layers.[20]

Figure 6 shows a schematic of a simple neural network and a deep neural network architecture, showing the key difference in the number of hidden layers. As most experts in the field have mentioned, there is no exact science or formula in the approach for selecting the number of hidden layers. The best and recommended approach has been to try a few, observe how they perform[19], and extrapolate from there with more testing and validation.

In this context of using DNN for decision making, we considered a few informational parameters: Pc, KLD, MHD, BD, MD, and the OA. We grouped the informational parameter into arbitrary assignments of 4 groups:

-Group 1 = {KLD, MD, BD, Pc, MHD}

-Group 2 = {KL, MD, MHD, Pc}

-Group 3 = {Pc, MHD, OA}

-Group 4 = {Pc}

We designed our DNN in MATLAB by defining the number of hidden layers, setting up the training, validation and testing ratios, and choosing the training functions. The typical recommended and considered ratio settings for training, validation, and testing were 0.7, 0.15, and 0.15 respectively[19]. We investigated two sets of DNN models that considered 10, 20, and 40 hidden layers for the variable sets of informational parameter groupings.

Backpropagation training functions

There are numerous backpropagation training functions that one can consider when designing a DNN. They can be grouped into 3 categories: (1) Backpropagation training functions that use Jacobian derivatives, (2) Backpropagation training functions that use gradient derivatives and (3) Supervised and Unsupervised weight/bias training functions[19]. In order for a specific application to use a DNN model for real-time applications, it very quickly becomes obvious that it is imperative that one understands the nature and quality of the data and the expected output, in order to decisively choose the right training functions to design the DNN model.

Jacobian Derivatives. The backpropagation training functions that depend on Jacobian derivatives can be faster but would then require more memory to work with and store the Jacobian matrices that can grow depending on the number of sources and weights. Examples of training functions in this category are the Levenberg-Marquardt (LM) and the Bayesian Regulation(BR) backpropagation.

In our application, we chose the LM backpropagation training function because of its promising performance and speed and it has an efficient implementation in MATLAB. The LM algorithm was designed to reach second-order training speed without having the need to compute the Hessian, which is a second order derivative matrix, or the derivative, of the Jacobian[21].

The LM algorithm uses the following approximation to the Hessian matrix ***H*** in a Newton-like update as shown in Equation (8) [22]:

$$w_{k+1} = w_k - \underbrace{[J^T J + \mu I]}_{H}^{-1} J^T e \tag{8}$$

where ***J*** is the Jacobian matrix that contains the first derivatives of the network errors with respect to the weights ***w*** and biases. ***e*** is a vector of the neural network errors, ***I*** is the identity matrix and ***μ*** is known as the combination coefficient[22,23].

Gradient Derivatives. The backpropagation training functions may not be as fast as the Jacobian backpropagation methods but they are appealing due to their potential support and implementation on a Graphics Processing Unit (GPU)as well as compatibility of running on the Parallel Computing Toolbox. There are numerous gradient derivatives algorithms to choose from; as long as the DNN model's weights and transfer functions have derivative functions, the gradient derivatives can be implemented[19].

In our application, we chose the scaled conjugate gradient algorithm due to its enhanced performance compared to other gradient derivatives methods[24]. During training, several stopping conditions need to be pre-defined such as the number of epochs (iterations), maximum duration, performance goal, minimum performance gradient, and validation performance[25]. Equations (9)-(13) capture the scaled conjugate gradient iteration algorithm, in which you are solving for the *N* weights ***x*** given the inputs ***b*** and ***A*** is a non-singular symmetric *NxN* matrix. The goal is to determine a search direction $\mathbf{p_k}$ over iterations ***i***, where ***i*** <***k***, and αk is the step size such that $x_k + \alpha_k p_k < x_{k+1}$[24,25].

$$Ax = b \tag{9}$$

$$r_k = b - Ax_k \tag{10}$$

$$p_k = r_k - \sum_{i<k} \frac{p_i^T A r_k}{p_i^T A p_i} p_i \tag{11}$$

$$x_{k+1} = x_k + \alpha_k p_k \tag{12}$$

$$\text{where,}\quad \alpha_k = \frac{p_k^T (b - Ax_k)}{p_k^T A p_k} = \frac{p_k^T r_k}{p_k^T A p_k} \tag{13}$$

Supervised and Unsupervised weight/bias training functions. For the supervised and unsupervised weight and/or bias training, the backpropagation functions are focused on the approach and order that the weights and biases are learned, trained, and updated. Examples include batch training, cyclical order, random order, and sequential order for supervised weight/bias training functions[19]. The unsupervised training functions include the batch training and random order approaches. These training functions were not implemented for our applications, however we will endeavor to explore them in our future work.

The following subsections will summarize the DNN using the three different enumerations of hidden layers {10, 20, and 40) and the two backpropagation training functions for the 4 different groups of information parameter sets. We used a sample data set of 1000 samples of simulated data containing both safe and close encounter classifications.

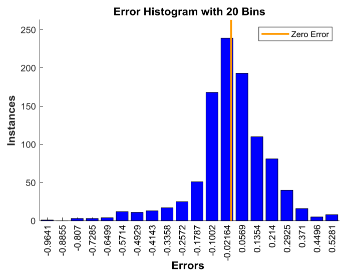
The performance will be measured using the regression R value which is an indication of the relationship between the outputs and the targets. If Regression is equal to 1, then there is an exact linear relationship between the outputs and targets; if Regression is equal to 0, then there is no linear relationship between outputs and targets. Regression=0 is what we will be using for this preliminary exploration, however there are other non-linear Regression constructs[26] that could be explored and may perhaps provide an improved result when the results are not constrained to a binary output {0,1} = {unsafe, safe}. *Note the binary outputs assignments for the DNN are not the same assignments as for the FIS, but bear similar theoretical meaning and representation*.

Results: Group 1 = {KLD, MD, BD, Pc, MHD}

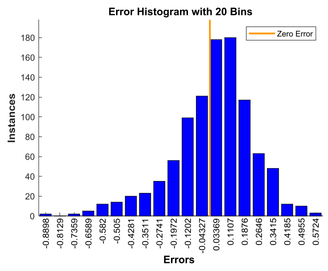
The SCG and LM training algorithm's results for Group 1 showed that the LM algorithm on the DNN with 40 hidden layers performed the best compared to the other constructs within the group as is shown in Table 2. An interesting observation to note is that the increase of the number of hidden layers does not necessarily imply an improved performance of the training algorithms for this particular grouping of information parameters. In Figure 7, the vertical red line shows the zero error location on the histogram plots. The SCG training algorithm's errors have a larger variance compared to the LM training algorithm. The errors are shown in Table 2 as the RMSE: Performance metric. Hence, a reasonable DNN model choice for Group 1 would be the LM training algorithm using 40 hidden layers. This would serve as an appropriate starting point to refine the DNN model's training requirements to further improve the performance and increase the Regression value.

Table 2. Performance Metrics for Group 1.

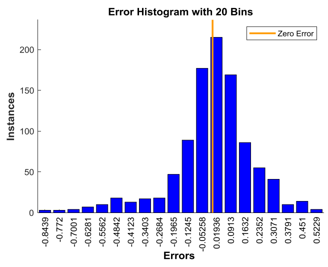
		Group 1		
		10 layers	20 layers	40 layers
Scaled Conjugate Gradient	Regression	0.91	0.89	0.91
	RMSE: Perf	0.0366	0.0469	0.0407
Levenberg-Marquardt	Regression	0.95	0.93	0.96
	RMSE: Perf	0.0232	0.0319	0.0192



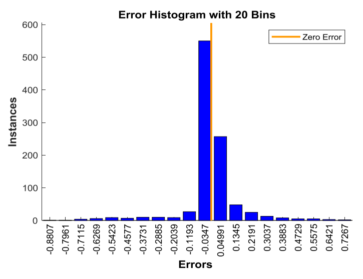
SCG 10 Hidden Layers



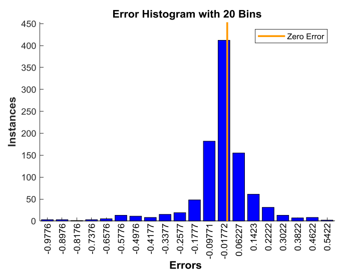
SCG 20 Hidden Layers



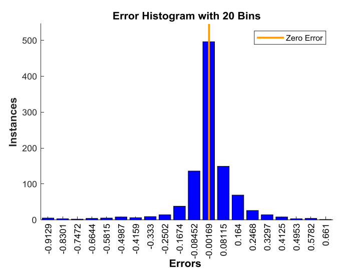
SCG 40 Hidden Layers



LM 10 Hidden Layers



LM 20 Hidden Layers



LM 40 Hidden Layers

Figure 7. Group 1 Error Histograms for the Scaled-Conjugate Gradient (SCG) *Top Row* **and the Levenberg-Marquardt (LM)** *Bottom Row* **training algorithms for the 10, 20, and 40 hidden layers. The vertical red line is the zero error mark.**

Results: Group 2 = {KL, MD, MHD, Pc}

Group 2 has a somewhat similar performance compared to Group 1 but with a slight improvement in the SCG training algorithm as shown in Table 3. The elimination of the BD information parameter could be a potential explanation on the improved performance of the SCG training method, as it is somewhat related to the MHD information parameter. We also see that the histograms exhibit tighter variances for Group 2 compared to Group 1 in Figure 8. A reasonable DNN model choice for Group 2 would be the LM training algorithm for all 3 categories: 10, 20, and 40 hidden layers.

Table 3. Performance Metrics for Group 2.

		Group 2		
		10 layers	20 layers	40 layers
Scaled Conjugate Gradient	Regression	0.93	0.91	0.92
	RMSE: Perf	0.0289	0.0369	0.0336
Levenberg-Marquardt	Regression	0.95	0.95	0.95
	RMSE: Perf	0.021	0.0209	0.021

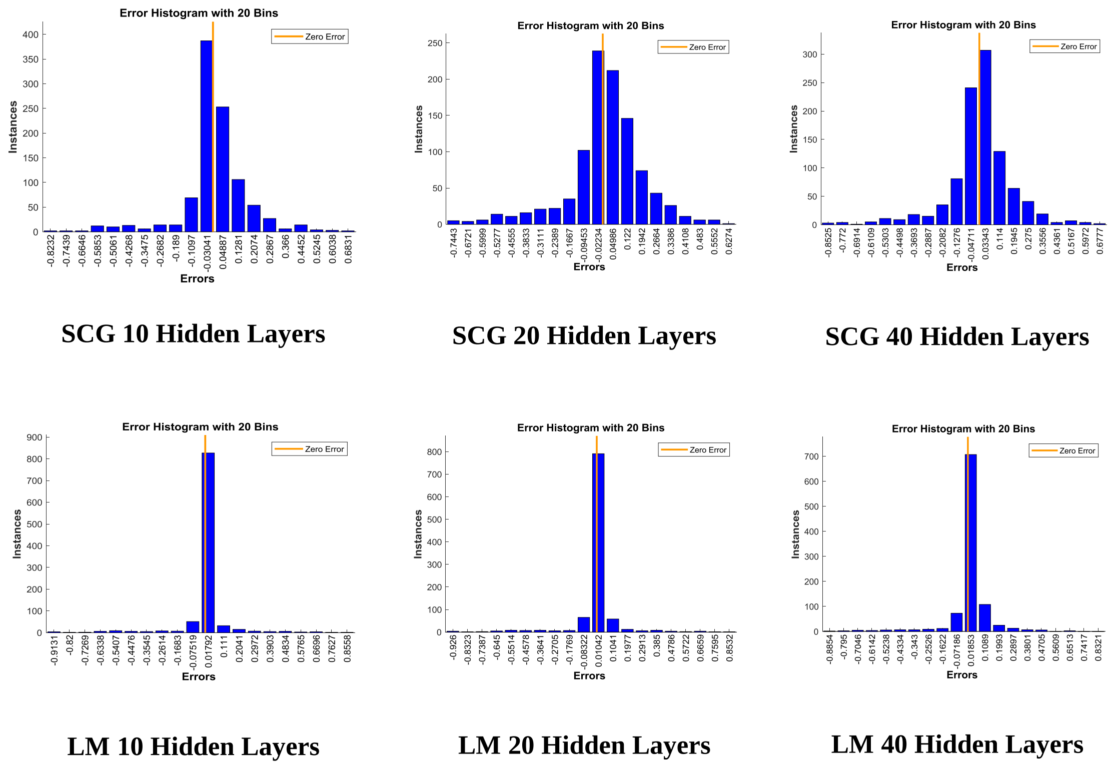


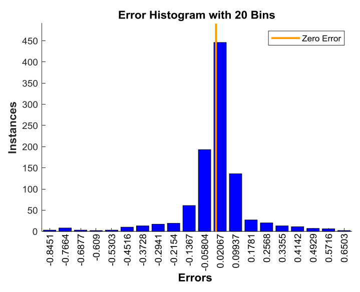
Figure 8. Group 2 Error Histograms for the Scaled-Conjugate Gradient (SCG) *Top Row* **and the Levenberg-Marquardt (LM)** *Bottom Row* **training algorithms for the 10, 20, and 40 Hidden Layers. The vertical red line is the zero error mark.**

Results: Group 3 = {Pc, MHD, OA}

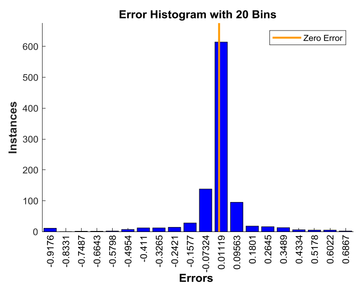
This is an interesting grouping that includes a geometrical information parameter (Orbit Angle between the Primary and Secondary object at TCA). This is the best performing grouping of all 4 groups considered with the best DNN model using the LM training algorithm and 10 hidden layers as shown in Table 4. Both the SCG and LM training methods have the lowest RMSE performance of all the 4 groups as seen in both Table 3 and Figure 9. We analyze the best DNN model in a little more detail below in Figure 10, to show how the DNN model can be validated, verified, and improved as a potential model for a decision making tool.

Table 4. Performance Metrics for Group 3.

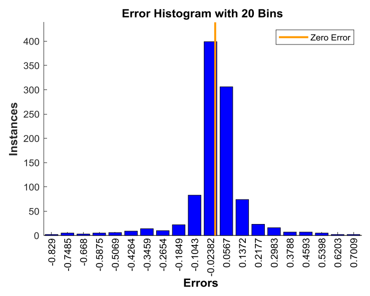
		Group 3		
		10 layers	20 layers	40 layers
Scaled Conjugate Gradient	Regression	0.93	0.93	0.94
	RMSE: Perf	0.0293	0.0284	0.0245
Levenberg-Marquardt	Regression	0.96	0.96	0.96
	RMSE: Perf	0.0177	0.0179	0.0179



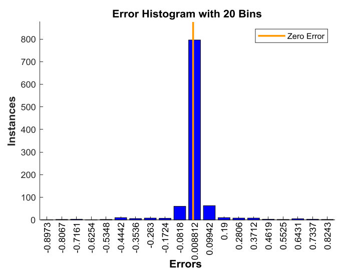
SCG 10 Hidden Layers



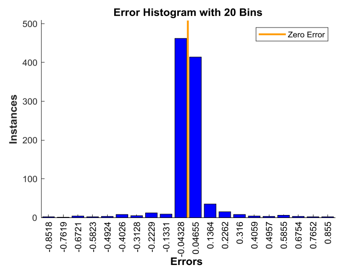
SCG 20 Hidden Layers



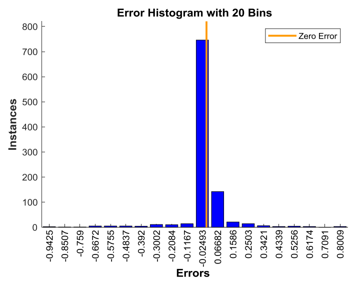
SCG 40 Hidden Layers



LM 10 Hidden Layers



LM 20 Hidden Layers



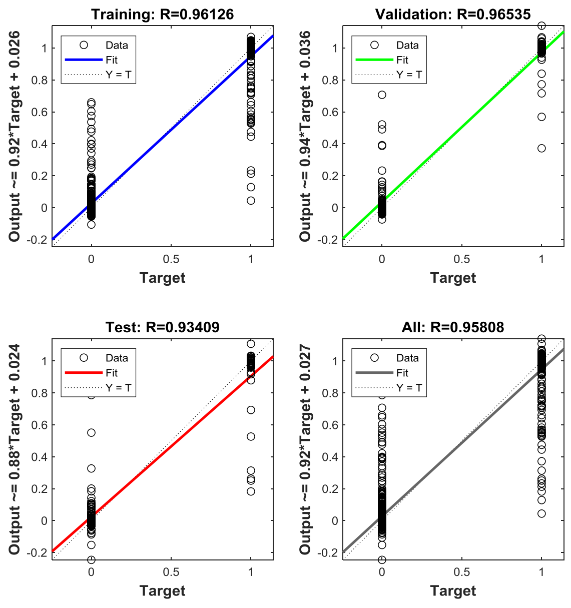
LM 40 Hidden Layers

Figure 9. Group 3 Error Histograms for the Scaled-Conjugate Gradient (SCG) *Top Row* **and the Levenberg-Marquardt (LM)** *Bottom Row* **training algorithms for the 10, 20, and 40 Hidden Layers. The vertical red line is the zero error mark.**

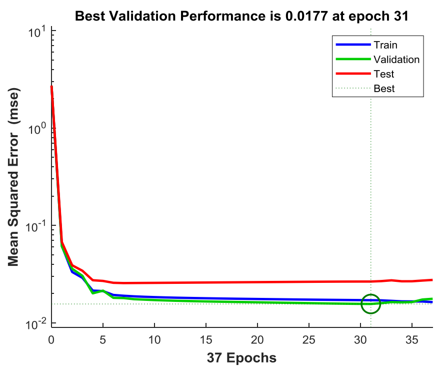
In Figure 10, the LM training algorithm with 10 hidden layers illustrates the Regression plot results as well as the epoch iterations performance plot. In the Regression plot, the data (70 percent of the available data) that was used for training the DNN model had a Regression performance of 0.961in comparing the targeted and output classifications. Fifteen percent of the data was used for validation and the other 15 percent was used for testing. Also both sets produced satisfactory Regression values of 0.965 and 0.934 respectively for an overall Regression value of 0.958.

The performance plot (Figure 10, *Right Plot*) is one of the visualization tools that can provide an insight into how the DNN model's training can be improved. The best validation training iteration epoch was at epoch 31, even though the DNN model continued to train until epoch 37. For this particular performance plot, the test curve did not deviate significantly after epoch 31 and seemed to taper-off at 0.0266 compared to the validation's curve performance of 0.0177.

Therefore, the DNN model can undergo an improved tuning to halt the epoch iterations at 31 and re-evaluated to observe the new performance. This proposed effort was beyond the scope of this current attempt but will be revisited for future work.



Regression Plot



Performance Plot

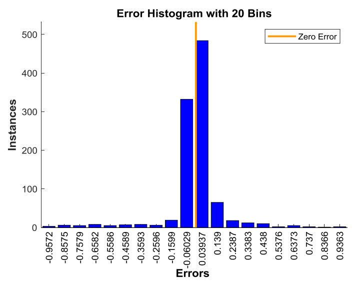
Figure 10. (Group 3) LM 10 Hidden Layers: Analyzing the DNN model performance after training and testing using the Regression and Performance Plots.

Results: Group 4 = {Pc}

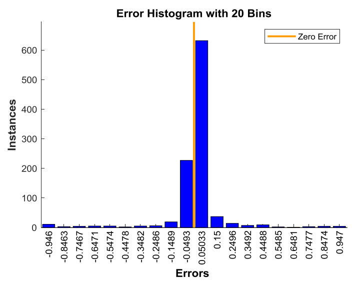
Lastly, we wanted Group 4 to be purely based on Pc alone as the current standard for conjunction analysis decision making. An interesting outcome as is shown in Table 54 is the virtually similar performance of both SCG and LM training algorithms. This can be expected due to the fact that the targets were determined by Pc thresholds alone for a binary result {0,1} = {unsafe, safe}. However, what is very interesting is that Group 4 is not the highest performing of all the groups despite Pc serving as the target determinant.

Table 5. Performance Metrics for Group 4.

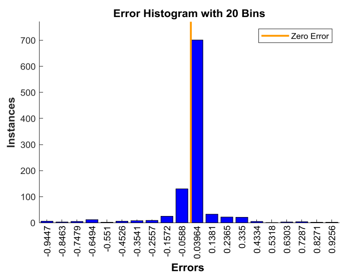
		Group 4		
		10 layers	20 layers	40 layers
Scaled Conjugate Gradient	Regression	0.93	0.92	0.93
	RMSE: Perf	0.0313	0.0342	0.032
Levenberg-Marquardt	Regression	0.93	0.93	0.93
	RMSE: Perf	0.0309	0.0303	0.0303



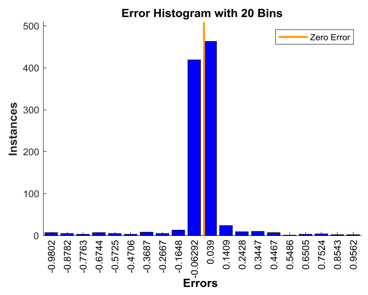
SCG 10 Hidden Layers



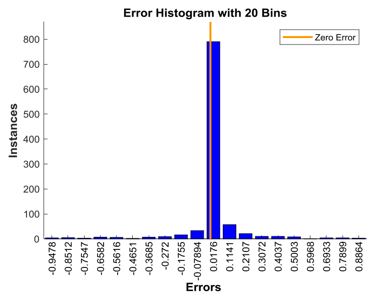
SCG 20 Hidden Layers



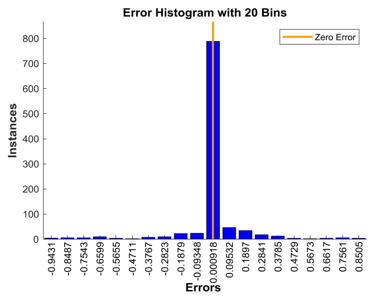
SCG 40 Hidden Layers



LM 10 Hidden Layers

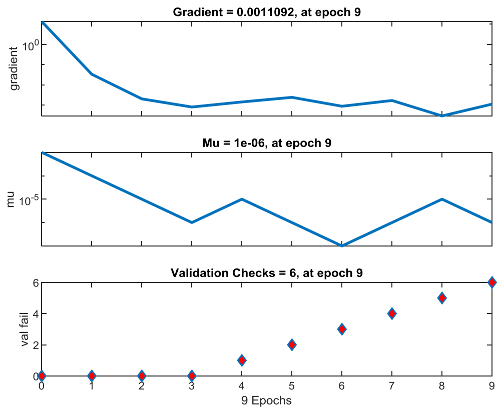


LM 20 Hidden Layers

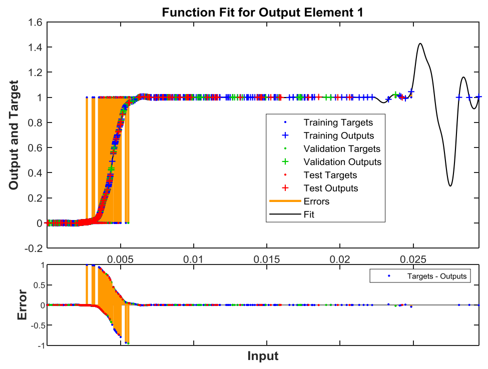


LM 40 Hidden Layers

Figure 11. Group 4 Error Histograms for the Scaled-Conjugate Gradient (SCG) ***Top Row*** **and the Levenberg-Marquardt (LM)** ***Bottom Row*** **training algorithms for the 10, 20 and 40 Hidden Layers. The vertical red line is the zero error mark.**



Training State



Output and Target Function Fit

Figure 12. (Group 4 LM 40 Hidden Layers) ***Left Plot*** **Training state outputs from the LM training algorithm.** ***Right Plot*** **Function Fit of Output and Target assignments with an error subplot.**

In Figure 12 the training state plots provide an insightful indepth surveillance that shows the gradient and $\boldsymbol{\mu}$ values' evolution of the LM training algorithm along the epoch iterations. In MATLAB, if the validation checks reach a default value of 6, the DNN model stops learning the data even if it was learning well from the data to train the DNN. Therefore, we can assume that the DNN was over fitted as you can observe the chattering of the gradient, $\boldsymbol{\mu}$ and Target-to-Output assignment plots.

In order to resolve this, a smaller number of hidden layers can be implemented as an investigation and potential solution to avoid overfitting.

CONCLUSION

In this task, we introduced an investigative approach for a potential fast and accurate close approach predictions based on artificial intelligence in areas of supervised, unsupervised machine learning and Deep Neural Networks. The goal for this research was to consider the study of statistical and information theory parameters in contrast and complementary to the classical probability of collision computation alone for conjunction analysis decision making.

The Deep Neural Networks presented a more promising path compared to the Fuzzy Inference System as a potential conjunction analysis decision making tool. However, ongoing research is underway to determine an optimal and representative physics-derived adaptive set of parameters for each conjunction case. We continue to retain the possibility of incorporating sensor tasking based on geometry for line-of-sight and observability as a constraint as an information metric that will be useful and crucial for the conjunction analysis task as a whole that incorporated orbit determination information.

ACKNOWLEDGMENTS

This work was funded by the NASA Goddard Space Flight Center (GSFC) Independent Research and Development Program. The researchers would like to thank Lauri Newman from the NASA Conjunction Analysis and Risk Assessment (CARA) Program for her support and Lavida Cooper from the NASA GSFC Technology Enterprise and Mission Pathfinder Office (TEMPO 450.2), as well as Jason Mitchell from NASA GSFC Mission Engineering and Systems Analysis' Assistant Chief Technologist.

REFERENCES

[1] Alfriend, K. T., Akella, M. R., "Probability of Collision Between Space Objects," *Journal of Guidance Control and Dynamics*, Vol 23, No. 5. September-October 2000

[2] Hall, D.T., Casali, S.J., Johnson, L.S., Skrehart, B.B., and Baars, L.G., "High Fidelity Collision Probabilities estimated using Brute Force Monte Carlo Simulations," *AIAA/AAS Astrodynamics Specialist Conference,* Snowbird, UT, August 2018

[3] Alfano, S. "Satellite Conjunction Monte Carlo Analysis," *AAS Space Flight Mechanics Meeting*, Pittsburgh PA, February 2009

[4] Frigm, R.C., "A Single Conjunction Risk Assessment Metric: The F-value," *AIAA/AAS Astrodynamics Specialist Conference Proceedings*, 9-13 August 2009

[5] Burton, A., Zielinski, M., Frueh, C., Mashiku, A., and Memarsadeghi, N., "Assessing Measures to reliably predict collisions in the presence of uncertainty," *AIAA/AAS Astrodynamics Specialist Conference* Proceedings, AAS 18-438, August 2018

[6] Frueh, C., "Realistic Sensor Tasking Strategies," *Proceedings of the Advanced Maui Optical and Space Surveillance Technologies Conference,"* Maui Economic Development Board, Maui, HI, Sept. 2016

[7] DeMars, K.J., and Jah, M.K., “Evaluation of the information content of observations with application to sensor management for orbit determination,” *Advances in the Astronautical Sciences*, pages 3169-3188, 2011

[8] Hershey, J.R., and Olsen P.A., “Approximating the Kullback-Leibler Divergence between Gaussian Mixture Models,” *2007 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 4, pages 317-320

[9] Mahalanobis, P.C., “On the generalized distance in statistics,” *Proceedings on the National Institute of Sciences of India*, Vol. 2, No. 1, pages 49-55, 1936

[10] Bhattacharyya, A. “On a measure of divergence between two statistical populations defined by their probability distributions,” *Bulletin of the Calcutta Mathematical Society*, Vol. 35, pages 99-109, 1943

[11] Bourbacki, N., “Topological vector spaces,” *Springe*r, ISBN 3-540-13627-4, Chapters 1-5, 1987

[12] Mathworks, “Applying Unsupervised Learning,” *The Mathworks Inc,*, Machine Learning eBook, Section 3 Version 5, 2016

[16] P. Torteeka et al., “Space Debris tracking based on fuzzy running Gaussian average adaptive particle filter track-before-detect algorithm”, *Research in Astronomy and Astrophysics*, 2012.

[17] L.P. Perera, J.P. Carvalho and C.G. Soares, “Fuzzy logic based decision making system for collision avoidance of ocean navigation under critical collision conditions”, *Journal of Marine Science and Technology*, March 2011.

[13] Mathworks, “Build Fuzzy Systems Using Fuzzy Logic Designer,” *The Mathworks Inc,*, https://www.mathworks.com/help/fuzzy/building-systems-with-fuzzy-logic-toolbox-software.html, 2019

[14] Sugeno, M., “Industrial applications of fuzzy control,” Elsevier Science Publishers Co., 1985

[17] Mathworks, “What is Sugeno-Type Fuzzy Inference,” *The Mathworks Inc,*, https://www.mathworks.com/help/fuzzy/what-is-sugeno-type-fuzzy-inference.html, 2019

[18] P. Long, W. Liu and J. Pan, “Deep-Learned Collision Avoidance Policy for Distributed Multi-Agent Navigation”, arXiv:1609.06838v1 [cs.AI] Department of Mechanical and Biomedical Engineering, the City University of Hong Kong, 22 Sep 2016.

[19] Mathworks, “Introducing Deep Learning with MATLAB,” *The Mathworks Inc,*, Deep Learning eBook, 2017

[20] Vazquez, F., “Deep Learning made easy with Deep Cognition,” *Becoming Human: Artificial Intelligence Magazine,* Dec 21, 2017

[21] Marquardt, D., “An Algorithm for Least-Squares Estimation of Nonlinear Parameters,” *SIAM Journal on Applied* Mathematics, Vol. 11, No. 2, pages 431-441, June 1963

[22] Hagan, M.T., and Menhaj, M., “Training feed-forward networks with the Marquardt algorithm,” IEEE Transactions on Neural Networks, Vol. 5, No. 6, pages 989-993, 1994

[23] Yu, H., and Wilamowski, B.M., “Levenberg-Marquardt Training,” Auburn University, Chapter 12, 2010

[24] Moller, M.F., “A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning,” *Neural Networks*, Vol. 6, pages 525-533, 1993

[25] Scales, L.E., “Introduction to Non-Linear Optimization,” New York: *Springer-Verlag*, 1985

[26] Fallah, N., et al., “Nonlinear Poisson regression using neural networks: a simulation study,” *Neural Computation & Application*, DOI 10.1007/s00521-009-0277-8, Springer-Verlag, 16 February 2008

[27] Chan, K., “Short-Term vs Long-Term Spacecraft Encounters,” AIAA/AAS Astrodynamics Specialist Conference, Providence, RI, August 2004