Rapid Missile System Characterization Using Computational Intelligence

by

Steven G. Ritz

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
August 2, 2014

Keywords: fuzzy logic, artificial neural network, missile

Approved by

Roy Hartfield, Chair, Walt and Virgina Woltosz Professor of Aerospace Engineering
Alice Smith, W. Allen and Martha Reed Professor of Industrial and Systems Engineering
Andrew Sinclair, Associate Professor of Aerospace Engineering

Abstract

There exist many physical relationships between dynamic systems that humans can infer data from where an exact science may not be possible or plausible to apply. The imprecision or inaccuracy of attempted models and sensors often lead to errors when seemingly simple tasks are left to a machine; a seasoned expert may know from experience how to react under certain conditions for which a robot may not have been trained or may not have sensors to a high enough precision. Fuzzy Logic and Artificial Neural Networks are an attempt to remedy that. If machine algorithms are able to infer solutions from previous knowledge, then new potential can be unlocked in computational applications. The study performed discusses the use of computational intelligence methods in the rapid classification of known and unknown missile systems. When a hostile missile is launched, it is crucial to know the geometry and capabilities of the impending ballistic vehicle. Computational intelligence mechanisms could play a large role in this defense application as well as a complementary tool to existing defense systems.

Acknowledgments

The author would like to thank Dr. Roy Hartfield for providing the opportunity, encouragement, and support for conducting this research. The author would also like to thank Dr. Alice Smith for the help and guidance with computational intelligence methods, without which the author's understanding of fuzzy logic would be fuzzy. Special thanks also goes to the author's parents and grandparents for their years of support and encouragement, without which none of this would be possible.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

η	learning rate for an ANN
μ	membership of a single observation
ν	cluster center coordinates
V	weight matrix between input layer and hidden layer
W	weight matrix between hidden layer and output layer
X_i	Input neuron for ANN
Y_k	Output neuron for ANN
Z_j	Hidden neuron for ANN
ANN	Artificial Neural Network
BP	backpropagation
c	index of cluster
d	distance from observation to current cluster center
FCC	Fuzzy Functional Clustering
m'	fuzziness parameter
SRM	solid rocket motor

Chapter 1

Introduction

Most anti-missile defensive measures rely on knowing the capabilities of a missile within seconds of launch. Often, the arsenal and capabilities of an adversary are known to the defensive group. More generally, the technological level of the missile system is characterized but a given missile systems capabilities may not be known *a priori*. There is therefore a need to quickly characterize hostile missile systems during launch. Many successful endeavors have been made to find the optimal missile configuration for a given trajectory using physical models [1, 2, 3, 4, 5]. Each of these methods requires the use of a population based meta-heuristic optimizer: genetic algorithms, particle swarm, ant colony, or a hybrid formed from a combination of one of these with a Hooke and Jeeves pattern search. Each member in the population of these methods was evaluated using a 6 degree-of-freedom missile code called AERODSN and compared to the objective function of the optimizer to determine fitness [6]. While these methods have been met with success, the amount of computational time required is not sufficient for response times if an unknown missile is launched. During this type of scenario, reliable information needs to be received within tens of seconds after launch if not sooner.

By using a method that requires a large up-front computational cost, faster calculations can be made. In this study an algorithm is proposed that uses a trained database of missile system trajectories to rapidly and accurately identify and predict the geometry of the missile during launch. The estimates are calculated using an artificial neural network (ANN) trained on simulated data representing possible missile systems and their trajectories. To improve the ANN performance, however, the training data was divided into subsets using a variant of the fuzzy c-means clustering algorithm, here referred to as Fuzzy Functional Clustering

(FFC). Separate ANNs were trained for each cluster of training data. This approach reduced the required domain for each ANN, allowing for improved accuracy within that domain. After training, each network undergoes a validation using data withheld from training in order to determine how well the network estimates the geometrical parameters from new data points.

A total of twenty data sets were used to characterize the entire method as a whole in addition to the validation metrics performed at the various stages of the algorithm. The results of the twenty trials were compiled for statistical use as well as to measure the accuracy and consistency of the algorithm.

1.1 Previous Work

1.1.1 Functional clustering

Functional clustering is not a new concept: a hybrid of the B-spline fitting and k-means was given by Abraham *et al.* [7] as well as Luan and Li [8] which clusters the coefficients of the spline polynomial, clustering around principal points of curves devised by Tarpey and Kinateder [9], Bayes clustering with hierarchical wavelets proposed by Ray and Mallick [10] (2006), and the others summarized by Chiou and Li [11]. Many of these use a variant of the k-means (also called c-means) clustering algorithm, but a fuzzy k-means algorithm is possibly better suited for this application. This assertion is based on how the fuzzy clustering algorithm calculates membership to a cluster, using not only how close it is to a single cluster but also the distance to all other clusters. Since the membership value is ultimately what decides which cluster to place an observation, the fuzzy clustering algorithm seemed to be a strong choice to couple with a functional clustering method.

1.1.2 Missile characterization

Other attempts have been made to characterize a missile in flight with different approaches. In 2005, Lei and Lu submitted a method for missile classification using micro-Doppler signatures from radar signals [12]. This method used the micro-Doppler signatures

from the radar in the time-frequency domain to classify targets based on their motion dynamics. This method appears to work well in determining the current performance of a missile in flight, but not the geometry. Their use of the Gabor feature extraction could prove as an improvement to the RMS translation as described in this study with some modifications; however, this would be the topic of a future study.

Another attempt to classify missiles in flight that used artificial neural networks was carried out by Albarado, Hartfield, Carpenter, Burkhalter, and Ritz [13]. In this study, the team used training data created by a Monte Carlo simulator for 6 reference missiles. Each missile was flown out for 10 simulations and then used in the training set to attempt to correctly classify missiles in flight. The initial training results were promising, but when validating with outside data generated by other simulations, the predictions started to misassign classifications. In addition, this work only attempted to classify, not characterize the geometry, of missiles.

This study is inspired from the previous motivation of earlier works, but attacks the problem with an alternate solution. Similar to the Lei and Lu method, a characterization attempt is made, but not by using the same type of data nor by using the same feature extraction. Likewise, an artificial neural network is used in this study with similar architecture to that of Albarado *et al*, but used for geometrical characterization instead of classification.

Chapter 2

Background

2.1 Fuzzy Logic

The concept of fuzzy logic was conceived by Lotfi Zadeh in 1965 [14]. He explained the need for a mathematical means of expressing the uncertain relationships between objects in the real world and the terms humans use to classify them. For example, a "tall man" or "warm weather" may hold different meaning between two individuals, but with fuzzy logic sets, a membership function or set of functions can be created that roughly translate these adjectives to crisp measurements. The best way to understand fuzzy logic is with an example of membership functions and rule sets. Figure 2.1 shows a plot of three basic membership functions. Each corresponds to a linguistic variable (terms such as "tall", "warm", "fast", etc.) and the degree of membership for that variable in a system.

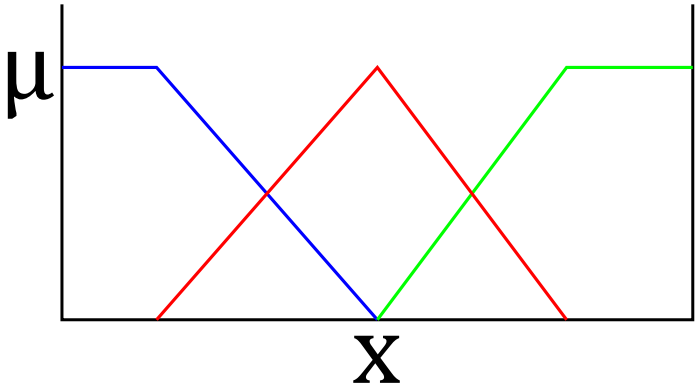
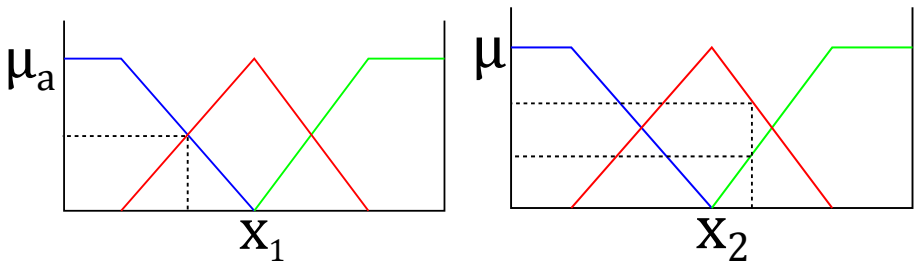


Figure 2.1: Example of a basic set of membership functions

Consider and example with two observations and one response. Each observation should have its own set of membership functions, which for this example is shown in Figure 2.2



(a) Membership functions for observation 1

(b) Membership functions for observation 2

Figure 2.2: Both sets of membership functions with marked observation x

The dotted lines in the figures represent the observations x measured for this example. Assuming that the range of membership values is $0 \leq \mu \leq 1$, μ_a for blue and red is equal to 0.5 while μ_b for red is 0.75 and μ_b for green is 0.25. Given that the activated rule set for this example is

IF R AND R, THEN B

IF R AND G, THEN G

IF B AND R, THEN G

IF B AND G, THEN R

the membership values for the response set of membership functions can be found. In this example, this is done using the max-min rule. The minimum membership value of all observations in a single rule is selected for the response variable. If multiple values for an output are calculated, the maximum value is chosen. The calculated response values are

$$\text{IF } \mu_{ar} = 0.5 \text{ AND } \mu_{br} = 0.75, \text{ THEN } \mu_{cb} = 0.50$$

$$\text{IF } \mu_{ar} = 0.5 \text{ AND } \mu_{bg} = 0.25, \text{ THEN } \mu_{cg} = 0.25$$

$$\text{IF } \mu_{ab} = 0.5 \text{ AND } \mu_{br} = 0.75, \text{ THEN } \mu_{cg} = 0.50$$

$$\text{IF } \mu_{ab} = 0.5 \text{ AND } \mu_{bg} = 0.25, \text{ THEN } \mu_{cr} = 0.25$$

where the notation μ_{ij} represents the membership function of color j in set i. It should be noted that the green membership function of the output was activated twice. Following the max-min procedure, only the green value of 0.50 can be used since $0.50 > 0.25$. With all

of the outputs for the activated rules found, the final output value is able to be defuzzified. The output values are first plotted on the response set of membership functions as seen in Figure 2.3

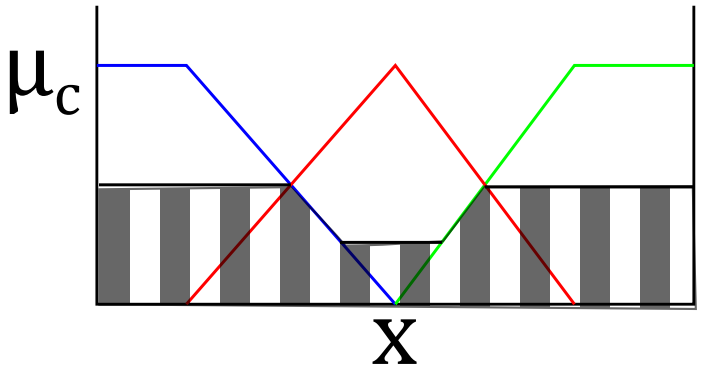


Figure 2.3: Shaded sections of the response membership functions corresponding to output from the rule set

These shaded areas represent the total membership for the outcome of the two observations. There are several acceptable methods for defuzzification, the most common being the centroid method. In this method the centroid of the shaded area is calculated. The x location of the centroid is the defuzzified value of the output.

The implications of this assertion of a method for handling the imperfections in measurements is not readily apparent unless one is acquainted with probability theory. The only other means of expressing uncertainty at the time was through probability, a subject fuzzy logic shares similarity but is not the same. For instance, the membership values should not be confused with probability; both concepts handle the idea of uncertainty in measurements, but probability is based on classical set theory and the *chance* a measurement is accurate [15]. Fuzzy logic membership explains the degree in which this measurement is correct. One could delve into the entire field of vagueness and uncertainty along with the impact of fuzzy logic, but the focus of this study is the concept of a membership function and its application to clustering.

The concept of clustering multi-variate data was also a fairly recent idea around the time fuzzy logic was introduced[16]. It took almost twenty years before the methods were

married by James Bezdek to form the fuzzy c-means algorithm, or FCM[17]. This proposed method coupled the notion of membership to the distance calculations associated with the c-means method. This seemed a natural addition as pattern recognition is integral to human perception just as much is the uncertainty in our speech [15]. The study presented in this text focuses on the application of the FCM in particular.

2.2 Artificial Neural Network

In general, an artificial neural network (ANN) is the result of an attempt to mimic a biological brain's ability to learn. Similar to how a brain is constructed, an ANN is composed of simple processing elements called neurons. These neurons can be connected in various ways by means of directed communication links [18]. The manner in which these neurons are connected determines the network's architecture. Simple architectures consist of feed-forward networks, those in which information flows through the network like a one-way street. There also exists more complicated architectures, like the cascade correlation which consists of adding in neurons to the topology until the error is reduced below a threshold. Both the feed-forward and cascade have a one-way operation, but other networks such as the bidirectional associative memory net can have connections that flow two-way.

The first ANN dates back to 1943 when McCulloch & Pitts published their paper titled "A Logical Calculus of the Ideas Immanent in Nervous Activity" [19], where they proposed a mathematical model for how a biological neuron was believed to work. In this early model, neuron activation was limited to binary functions: activated or not activated. Over the years, more sophisticated learning methods, the methods by which the weights that connect a network are changed to produce the correct values of outputs, were developed to handle more complex problems. After becoming very popular in the 1950s and 1960s, ANN research declined as current methods at the time could only solve linearly separable problems as shown in a report by Minsky and Papert [20]. In the 1980s, ANN research saw

a resurgence of activity after the publication of several new learning methods for networks such as backpropagation [21] and Hopfield learning nets [22].

For most architectures, the structure of a neuron is similar; it consists of an activation or transfer function that determines the type of signal a neuron fires with connections to the inputs and outputs. The basic structure of a neuron is shown in Figure 2.4.

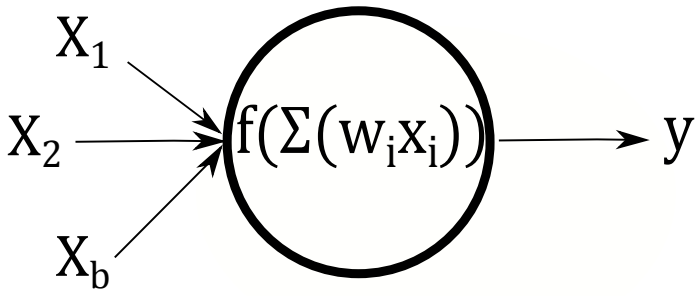


Figure 2.4: Example of basic neuron with activation function

In the previous figure, x is the raw input to the neuron, w_i is the weight coefficient to the corresponding connection, and y is the output after passing through the activation function, f. The x_b is an input referred to as the bias. It is usually equal to unity and when multiplied by w_b acts as a constant for the network. A bias is not required in neural networks but can be useful for adding some linearity to a model [18].

Early neuron models had binary transfer functions, allowing a neuron to send only "on" and "off" signals. This severely limited the type of problems ANNs could handle. In modern neuron structures, a sigmoidal transfer function σ is often used which is given as

$$\sigma(x_i') = \frac{1}{1 + \exp(-x_i')} \tag{2.1}$$

where x_i' is the input received by the neuron after multiplied by the weight matrix, W, as given by

$$[X]\,[W] = X' \tag{2.2}$$

This weight matrix is where the information of a network is stored once trained. It is what defines two unique networks with similar architecture and neuron structure.

A full network is composed of interconnected neurons. Different architectures have different methods for connecting the neurons but a common one is the feed-forward topology. This is composed of at least least three layers in the network: the input, hidden, and output layers. A very simple feed-forward network is shown in Figure 2.5

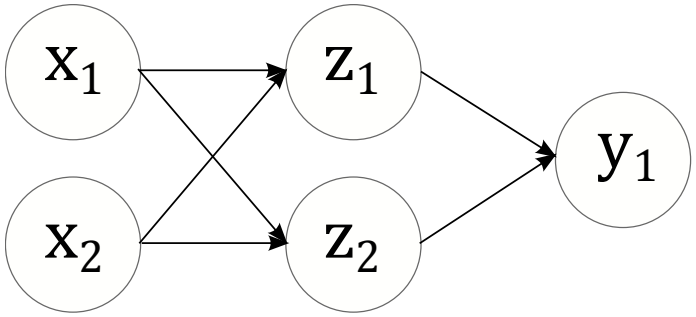


Figure 2.5: A 2:2:1 feed-forward network

where the input layer is represented by the x, the hidden layer by the z and the output by the y. Each connection is where the previously mentioned weights are stored; the output of each neuron is multiplied by the weight in the connection before being passed into the neuron on the path.

The basic principle of training a network is to minimize the error in the weights, but there are various techniques to accomplish this. Many are based on gradient methods, such as backpropagation, meaning they follow the downward slopes in the error function. A shortcoming of gradient methods, though, is the likelihood of ending up in a local minimum rather than in the global minimum. For most well-behaved data sets this is not a large problem, but for ill-behaved data this can cause poor training.

Chapter 3

Algorithm Development and Discussion

The full algorithm, as used in this study, can be described in five general steps:

1. A database of missile performance data is generated.

2. An initial analysis on data to select a reference missile is performed.

3. Trajectories are clustered based on the reference missile.

4. The ANNs are trained based on the associated clusters.

5. The ANN and clustering are evaluated by simulating a real-time flyout.

A more comprehensive discussion of the data sets is given in Chapter 4. This chapter is more focused on the computational intelligence methods of the study.

3.1 Fuzzy Functional Clustering

Clustering is a method for assigning labels to unlabeled observations in multi-dimensional space [23]. Through clustering the underlying patterns between observations are used to place the data in various sized groups that have similar characteristics. In the algorithm known as the c-means method, clustering occurs through an iterative process of calculating the cluster centers and the distances between the observations and the centers. For most clustering methods, the number of clusters is determined before implementing the clustering algorithm. For this reason, several iterations of clustering may be necessary to determine the optimal number of clusters for the application.

The fuzzy c-means (FCM) algorithm implemented in this process was established by Bezdek, Ehrlich, and Full *et al.* in 1984 [17]. It is a variation of the k-means clustering algorithm that incorporates a membership calculation that updates the centers of the clusters. This membership function is given as

$$\mu_{ik}^{t} = \left[\sum_{j=1}^{c} \left(\frac{d_{ik}^{t}}{d_{jk}^{t}} \right)^{2/(m'-1)} \right]^{-1} \tag{3.1}$$

where μ is the membership value for observation k in cluster i, d is the distance between the cluster center and the observation, t is the current iteration, and m' is the fuzzy parameter, which in all trials run using this process the fuzzy parameter was set to 2. This means an observation's membership is dependent on its' distance to all the cluster centers. The equation for the cluster center, ν, calculation is shown in the following

$$\nu_{ij} = \frac{\sum_{k=1}^{n} \mu_{ik}^{m'} x_{kj}}{\sum_{k=1}^{n} \mu_{ik}^{m'}} \tag{3.2}$$

which is similar to the hard k-means center calculation except for the membership values used as weighting coefficients in the calculation. After the center is calculated, the membership values are re-evaluated for the observations. These new values are then used to determine the new cluster center positions and so on until convergence or until the centers move less than some threshold value.

In the c-means clustering, the process of updating cluster centers is carried out until convergence; however in the FCM convergence is not always necessarily achievable. It is more practical to iterate until cluster centers change within a tolerance established or to continue over a set number of iterations.

The entirety of the FFC is listed for reference and can be found in Appendix B.

3.1.1 Selection of Reference Missile

The reference missile serves as the "truth" for calculating the RMS values (explained in the following section) that will be clustered. Selecting a good reference missile is a crucial task; if a sub-optimal missile is chosen the resulting cluster space will not yield even clustering meaning too many trajectories will end up in a single cluster. To prevent this, a method was found that provides two candidates for a reference missile in a given data set.

The maximum range and altitude are found within each set of missile trajectory observations in the data set. The mean and standard deviation of these values are then calculated. The first reference missile candidate is simply selected by finding the associated trajectory with the maximum altitude. The second candidate is selected by finding the trajectory with the maximum range at least one standard deviation less than the mean. Both missiles are used in the initial clustering and evaluated by calculating the standard deviation of the number of missiles in each cluster. For example, if candidate one is used in the FFC and four clusters are produced with population 1, 1, 2, 9, respectively, and candidate two makes the FFC produce four clusters of 3, 3, 4, 3, candidate two will be selected as the better missile to proceed with for the rest of the algorithm.

Alternatively, a method for choosing a reference missile is to run the batch clustering algorithm over as many iteration as there are missiles in the database, using every missile as reference missile. This will allow the user to visually inspect the cluster space generated by each choice. A better clustering might result from this method, however it is time consuming for both the computer and the user. For that reason the previous method presented is the preferred method of selecting a reference missile.

3.1.2 Translation into RMS Coordinates

Each trajectory that was clustered is made of hundreds or thousands of data points from incremented time-steps. This presents a unique challenge since the objects that were to be clustered were the trajectories themselves. A form of feature extraction needed to

be performed on the data points that represented the trajectories. This was accomplished by translating the full altitude, range, and velocity profiles of a trajectory into a single RMS value of the residuals as compared to the reference missiles chosen. Essentially, each trajectory would then be able to be represented as a point in 3-D space if plotted on axes RMS altitude, RMS range, and RMS velocity. These values were calculated using Equation 3.3 [24]:

$$\text{RMS} = \frac{\sqrt{\sum (y_t - y)^2}}{n} \tag{3.3}$$

where y_t is the target found from the spline, y is the data point from the missile in question, and n is the total number of data points given in that trajectory.

Other forms of feature extraction seem plausible for this type of clustering, however, the seemingly obvious choices fall short when applied to the classification section of the algorithm. For example, based on the type of data being clustered, it may seem sufficient to cluster this data based on just the max altitude, max range, and max velocity or some combination of those features. This falls short in the classification though due to the fact that the maximum aspects are not known *a priori* or as radar data is streaming in. While this method of clustering might work well for the unsupervised clustering of the initial database sets, its capabilities are lacking in the supervised classification. The method presented can be used for both scenarios with slight modification.

3.1.3 Selecting values for constants to be used

The values of the three constants of the FFC should be investigated over the course of any future study that wishes to implement this algorithm. These values can alter the type of clustering that will result from the FFC. The number of iterations for cluster center updating will likely scale with the size of the data set. Forty iterations was found to be suitable for data sets under 200 missiles. Selecting the number of clusters, c, can take some trial and error to find the optimal number that suits the user's specifications for clustering,

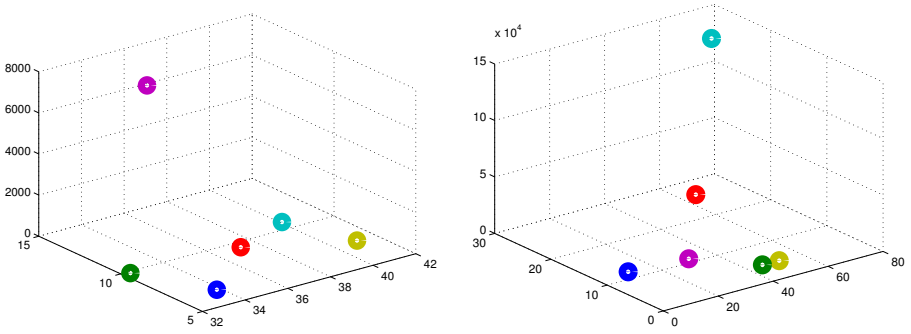
but it was found that a substantial number of clusters can scale with the size and variability of the data set. Too many clusters will lead to the formation of singletons, or clusters with a single observation; not enough clusters will result in clusters that missiles with too dissimilar of qualities. It is suggested to start with 6 or 8 clusters for data sets between 100 and 200 missiles that were generated over a uniform distribution. The fuzzy parameter, m', determines how "fuzzy" the cluster membership is. Typically this is set to 2 and adjusted as needed.

3.1.4 Initialize membership values for all missiles

The initialization of the membership values was simply done by systematically assigning a unity membership to every kth missile such that each cluster had roughly an even distribution of missiles. While the initial membership assignments may effect the final cluster assignments, it does not effect the shape of all the clusters. For example, if missile 1 and 2 are assigned to cluster 1 with one type of initialization and cluster 2 with another, there is no effect on the final outcome of the algorithm as long as those two missiles are clustered together.

3.1.5 Calculation of center

The cluster center calculation in Equation 3.2 is used for each cluster using the memberships of each missile and the coordinates: RMS altitude, RMS range, and RMS velocity. Initially these centers are located arbitrarily due to the seeded membership values. With each updated membership and center calculation, the centers shift resulting in the formation of clusters. Though the centers with never truly converge, their change in position will approach zero. If upon inspection these centers still move substantially after the maximum number of iterations is reached, increase the maximum number of iterations. It is crucial these centers fall into stable locations. Figures 3.1a and 3.1b show how much cluster centers can change over the course of a full set of iterations.



(a) Cluster center initialization (b) Cluster center final positions

Figure 3.1: The initial center positions vs. the final positions. This is meant to demonstrate the how the cluster centers move over iterations of cluster updates.

3.1.6 Distance and membership calculations

Euclidean distances from represented missile points to the cluster centers are found by using the RMS coordinates. These distances are then used to calculate the membership values for each observation. This is performed with Equation 3.1. Each observation is then assigned a cluster determined by the highest corresponding membership value obtained in the current iteration. Traditionally this is performed by implementing what is called an alpha cut, a method that handles cluster assignment by restricting acceptance to certain membership levels. A tournament selection was chosen over this method due to some errors that appeared in the classification section when implementing the alpha cut. This means that whichever cluster the observation showed the highest membership in was the cluster the observation was assigned to. The errors from the implementing the alpha cut are discussed later in the results given in Section 4.2.1.

3.2 Feed-forward Backpropagation Network

Once each data set that is passed through the initial clustering, the full trajectories of each missile are sorted into sub-sets based on their cluster designation. These sub-sets then act as separate training sets for the artificial neural network (ANN) for each cluster. For example, if the desired number of clusters is 6, then 6 ANNs will be trained to represent

each cluster. Instead of creating a single network trained on the entire data set, the networks were separated in this manner in order to train on more localized ranges of the geometrical parameters. If the training data range is too large, some resolution could be lost in the training process.

The specific type of network used for this algorithm is a single hidden layer, feedforward, backpropagation artificial neural network. This means the inputs are fed to a hidden layer and then to the output layer in a one-way process. The weights are reduced using a backpropagation method explained in further detail in the following section.

3.2.1 Backpropagation

The backpropagation algorithm is a an optimization method for the training of the weights in a feedforward network. The training is based on the idea of gradient descent of the error, meaning a weight change that produces a decrease in the error will be taken and any increase in the error stops the training. The basic architecture of a backpropagation network is shown in Figure 3.2 where the inputs to the network are given as X_i, the response

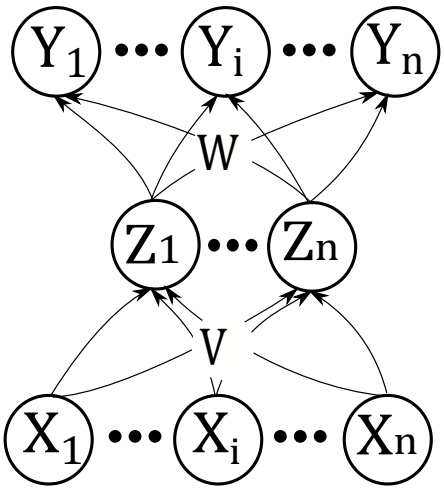


Figure 3.2: Sample backpropagation network architecture

variables are shown as Y_i, and the hidden layer neurons are given as Z_i. The weights connecting the first and second layer are V and W, respectively. These weights are the elements of the network that are trained using target data from the training set. For the

application presented in this study, this was the trajectory data. These weights are randomly seeded initially and then evaluated with the first observation selected from the training set following the same evaluation rules as any other feedforward architecture. Once the initial evaluation is complete, the weights are updated to minimize error. The backpropagation method starts with the output values and works back through the network, updating the weights in each layer *simultaneously*. It is important that the updated weights W are not used in the calculation of the updated weights V. The weight change for the individual weights in W when using a bipolar sigmoid activation function is given by the following

$$\Delta w = \eta(t - y)(1 + y)(1 - y)z \tag{3.4}$$

and the weight change for the weights in V is given by

$$\Delta v = \eta \left(\sum_k ((t - y)(1 + y)(1 - y)z)_k w_k \right) (1 + z)(1 - z)x \tag{3.5}$$

where η is the learning rate of the network training, k is the number of outputs, and t is the target data point the network is training to. This process is continued until the error in the network is smaller than some tolerance or until a maximum number of iterations has been reached. Once training is finalized for the network the weights are frozen and saved to be used for predictions. With each network undergoing training, a subset of the data to be trained with was withheld from actually being used. By testing the network with a data set that has not been used to train it, the overall accuracy of the network can be determined. If a poor prediction results from testing the with the holdout data, it is likely the network has been over-fitted. This means that particular set of weights found will predict the target data very well but will poorly predict any other data even if it is within the range of the originally trained data.

3.3 Classification/Prediction

The classification and prediction section of the algorithm is where the data gathered and trained from the previous sections is applied to characterize a missile while in flight. In the classification portion, the the radar data is read in and compared to the same reference missile used in the previous FFC with the same data set. This translates the newly received data into RMS coordinates. The distance from the observation to each predetermined cluster center is calculated and then used to calculate the membership using the same Equation from 3.1. Where this varies from the FFC is that the cluster centers do not get updated; they are fixed in space. The missile associated with the observed data is then assigned to a cluster based on the highest membership calculated. This cluster designation determines which weight matrix to use for the ANN prediction. For example, if a missile is assigned to cluster 3, then the weight matrix found from training the data associated with cluster 3 will be used to predict the desired characteristics. This process continues for every observation that is read in.

For a full application of the algorithm, the previous sections do not need to be run more than once; once the data is clustered and trained within acceptable error tolerances there is no need to re-run. The following chapter outlines how this algorithm and process were used for a validation study which also serves as an example for the use of this package.

Chapter 4

Application of Algorithm

4.1 Description of Data Sets

Apart from the main goal of predicting the geometry of a missile in flight, another goal of this application was to show the value of clustering a data set prior to training an ANN and to validate the accuracy of this method. In order to accomplish this, twenty data sets were created using the SRM flyout code and seeded using a Latin hyper-cube uniform distribution. Each of the unique missiles had a single trajectory assigned to it. The ranges for the 35 parameters were equal across all twenty data sets and are shown in Appendix A. The desired parameter to match in this study was diameter of the missile; therefore, many of the other parameters and characteristics of the missile were held constant or to a small range (some values, like star points, needed to be varied in order for a feasible solution to be found); however, the same fuel and launch angle were used in each simulated flyout as well as payload. If these certain values were not held constant, the range and altitude would be far less correlated with the diameter of the missile. For each unique missile created a set of data points were created to simulate incoming radar data. The number of these data points varied based on the flight time of the missile. Each data set contained 125 to 135 missiles with 3 missiles that were placed in each of the 20 data sets: a short range missile, a long range missile, and a mid-range missile. These three missiles were used to validate the accuracy and confidence of the total algorithm. Figures 4.1, 4.2, and 4.3 show models of the missiles used in this study.

The diameter was chosen for characterization due to the fact that most of the missile geometry is scaled to the diameter; it is the one parameter that can tell the most about a

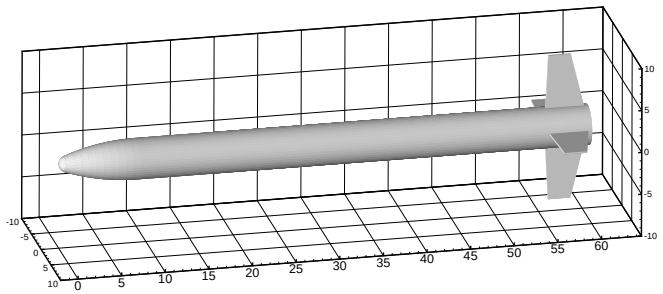


Figure 4.1: Long range, high altitude trajectory missile

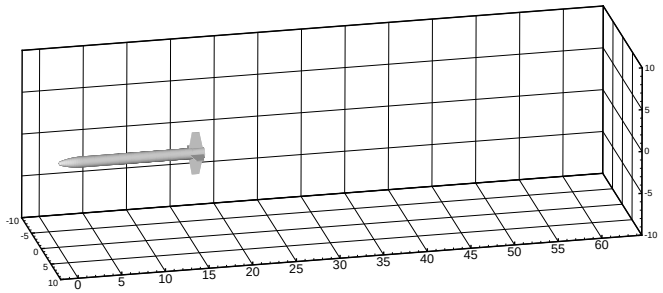


Figure 4.2: Short range, low altitude trajectory missile

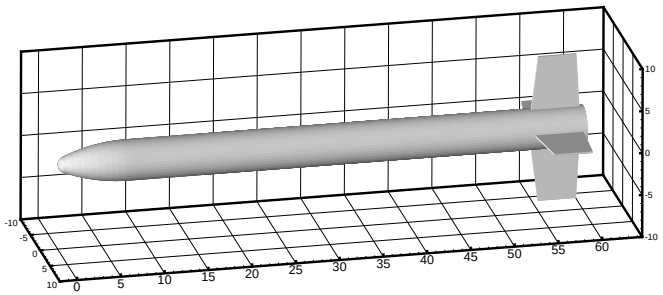


Figure 4.3: Mid range, mid altitude trajectory missile

missile. This not only reveals vital information about the geometry but the performance of the missile as well.

4.1.1 Selected parameters for clustering and classification

Four parameters were chosen to be the input for the FFC algorithm: time, altitude, range, and velocity. These were chosen to simulate the type of data that would be obtained by radar. The altitude is defined as the total height above sea level and the range is defined as the component of the distance from the launch point to the missile projected on the ground. The velocity is simply the time rate of change in position of the missile. Though there is not a direction associated with this velocity term in the data, it is implied that this term is tangent to the point of the trajectory where the measurement was taken or simulated. Though more data might have yielded better results, the need to simulate data strictly from a radar limited the number of plausible inputs to time, position, and any derivatives associated with those.

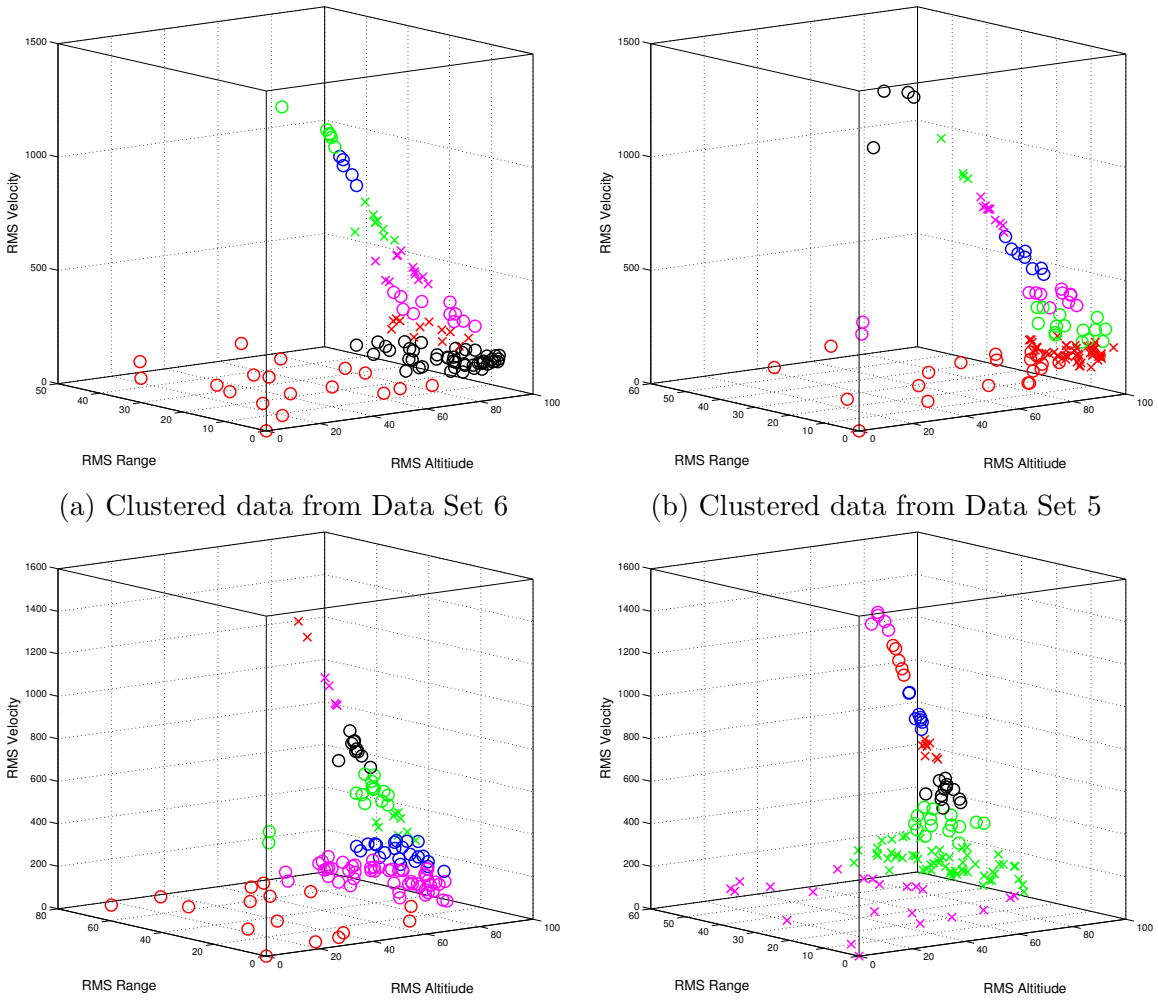
Since the classification is designed to perform in real-time with a missile launch, the input variables needed to be adjusted. The critical time for classification is in the early part of the flight. Early after launch, many missile trajectories can seem similar; many will have similar altitudes and ranges. For this reason, the classification only uses the velocity data from the radar data.

4.2 Results

The following sections are broken into the different parts of the algorithm and describe the validation and analysis of through each step. The final code was combined after all of the individual sections functioned properly and produced favorable results.

4.2.1 Functional Fuzzy Clustering Results

The results from the FFC in Figures 4.4a-4.4d are represented in the translated RMS coordinates for ease of viewing. These observations have no real bearing in physical space, they are simply a representation of the features extracted from the trajectories.



(a) Clustered data from Data Set 6

(b) Clustered data from Data Set 5

(c) Clustered data from Data Set 19

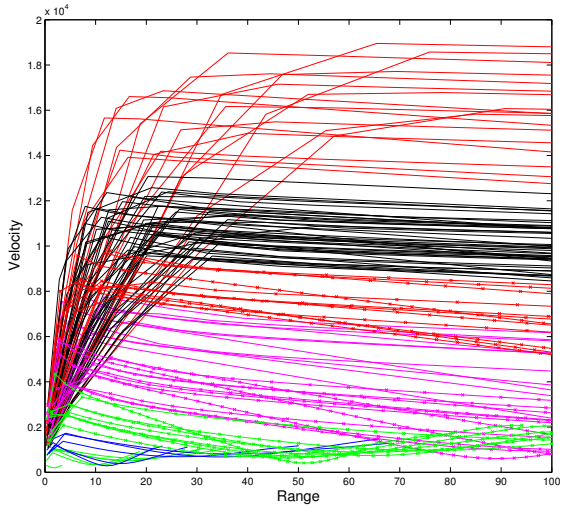
(d) Clustered data from Data Set 15

Figure 4.4: Selected RMS plots showing cluster patterns among data sets

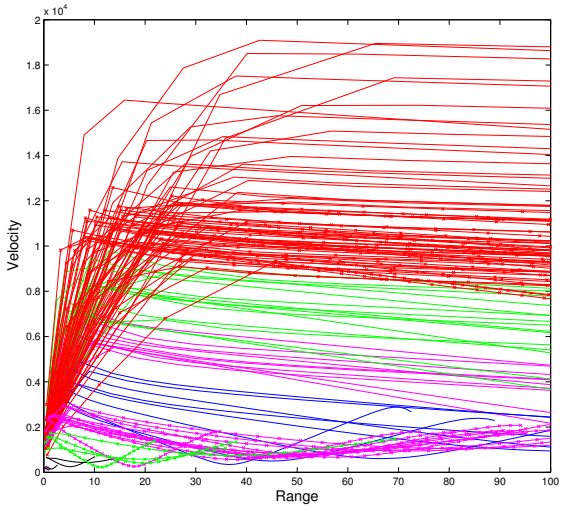
From these plots it is also easy to see the large differences in velocity and how that affects the clustering outcome. This result helps defend the reasoning to use just velocity as the classification variable during the characterization algorithm and also confirms the distinct variability in the missiles of the data sets; this implies that that some missiles share similar

trajectories but vary in velocity. This difference is only achievable with different size motors (since all missiles had the same fuel/binder type), which is only possible if the diameters vary, further implying that velocity was a good input for the classification.

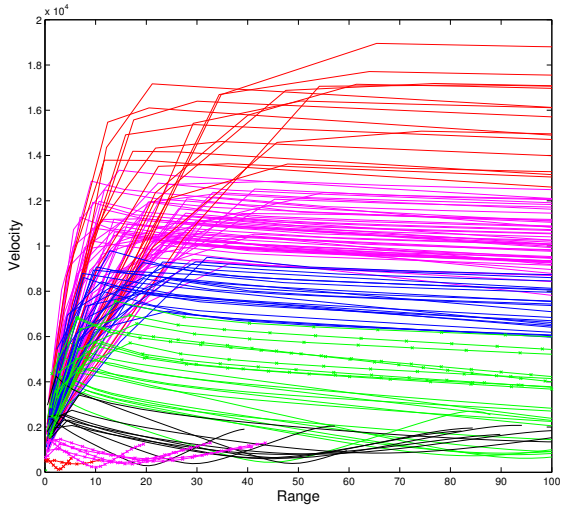
Figure 4.5 gives a more physical sense of how the clusters are grouped. Since the maximum ranges vary wildly in the data set, only the first 100,000 ft is displayed for ease of viewing the smaller missiles.



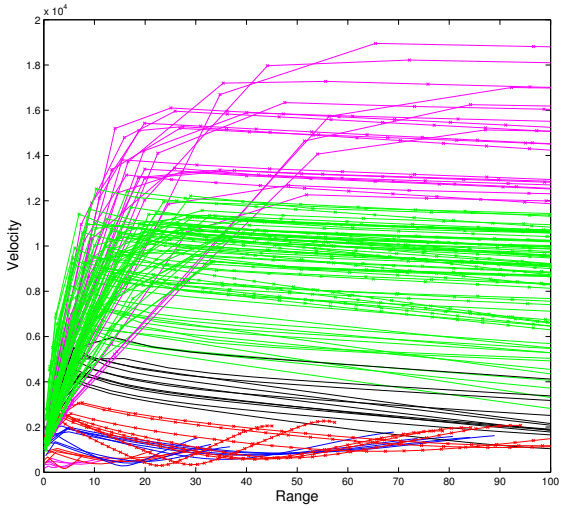
(a) Clustered trajectories from Data Set 6



(b) Clustered trajectories from Data Set 5



(c) Clustered trajectories from Data Set 19



(d) Clustered trajectories from Data Set 15

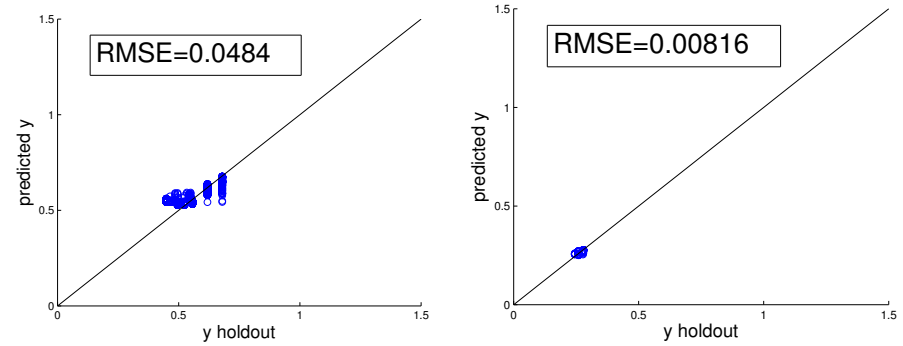
Figure 4.5: Selected RMS plots showing cluster patterns among data sets

The clusters seem to form bands along the velocity axis and are based heavily on the velocity after the motor burns out. The high density of clusters in the initial stages is found to cause some issues in classification that is further explained in Section 4.2.3.

It should also be mentioned that these clusters are the result of using a tournament based cluster assignment previously remarked in Section 3.1.6. This was used to ensure that each missile was placed into a cluster and no dead clusters formed. When using the alpha cut method, there were several missiles that were placed into singleton groups. Normally, a simple fix might be to increase or reduce the number of clusters in order to accommodate this outliers. Instead, the tournament selection was chosen because of the real-time classification operation. If the clustered data were to be used to classify another batch of trajectories this would not have been an issue, but with the nature of a moving time scale and the necessity of having an assigned cluster each time step, the missile needed to be assigned to the cluster with which it shared the highest membership. Future versions of the FFC may not involve this type of cluster selection but this was the most direct method for this application.

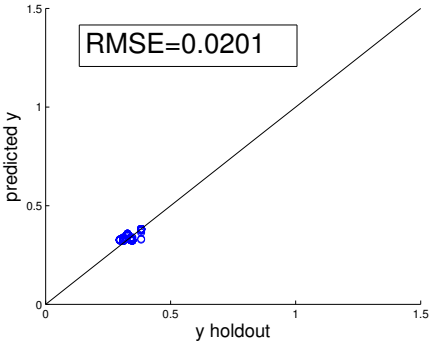
4.2.2 ANN Results

All networks were trained for a maximum of 2000 iterations or until the absolute error converged below 0.05, although in every scenario the maximum number of iterations was reached. As mentioned previously, the maximum number of iterations was limited to prevent over fitting. Each network underwent a validation performed by using a holdout data set created from pseudo-randomly selecting 20% of the observations; the three missiles in question were ensured to be placed in the holdout set. Figures 4.6a-4.6h show sample correlation plots from one of the data sets created from the results of using the inputs of the holdout data to make predictions. A dot on the diagonal line means the output was perfectly predicted after passing the inputs through the network. The RMS error is given for each validation.

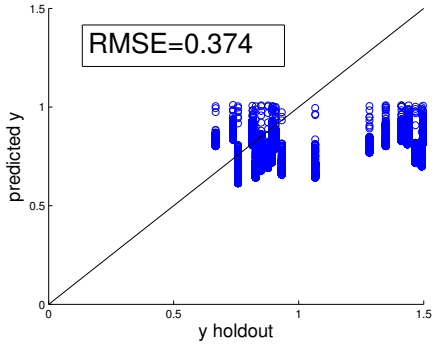


(a) Cluster 1 validation results

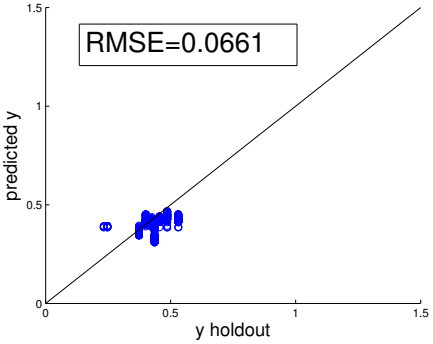
(b) Cluster 2 validation results



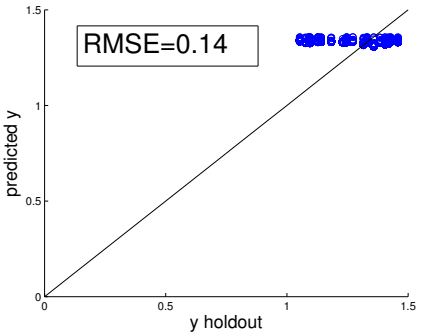
(c) Cluster 3 validation results



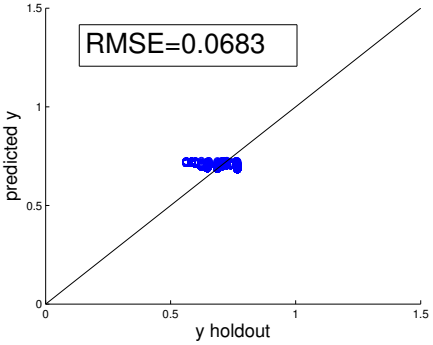
(d) Cluster 4 validation results



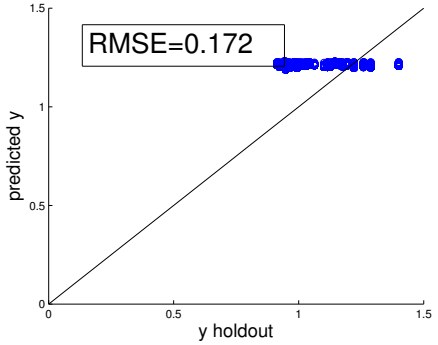
(e) Cluster 5 validation results



(f) Cluster 6 validation results



(g) Cluster 7 validation results



(h) Cluster 8 validation results

Figure 4.6: Correlation plots showing the calculated diameter results, in meters, of the ANN training validation with 20% holdout. Observations closer to the line $y = x$ are better predictions.

In this figure, Cluster 4 was visibly the worst performing cluster. It is important to note that though the RMSE seems very low, this type of result is unusable due to the poor correlation. This may have been due to the large number of missiles placed in this cluster relative to the size of the other clusters in this validation case. Increasing the number of clusters may fix this issue, however this action would likely result in dividing all clusters including the already small ones.

To demonstrate the significance of training on the subsets created by clustering, a similar training was performed on a holdout set of the entire data set. The results of this training can be found in Figure 4.7.

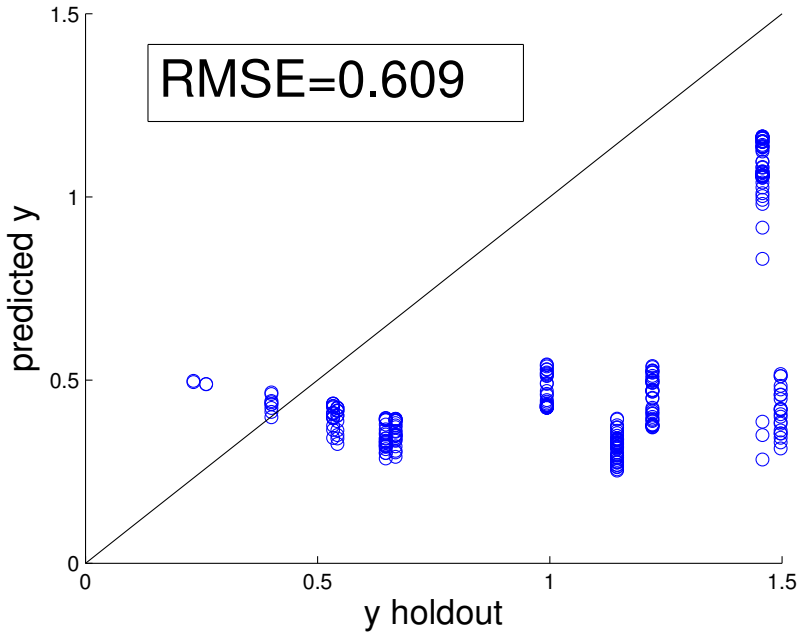


Figure 4.7: Correlation plot of the validation of a single artificial neural network representing the entire data set. Observations closer to the line $y = x$ are better predictions.

Given the same training parameters as the networks formed from the clustered data, the validation of this network shows a weak training result. This may have been due to the training set covering too wide a range of diameters for the network to properly obtain a higher degree of resolution.

4.2.3 Characterization Results

The results discussed in this section are the averages obtained from the same three missiles in the twenty data sets, averaged together, then plotted to determine if the predictions fell within the bounds of an acceptable error. In the following cases, an error of ±5% was determined to be acceptable. Figure 4.8 show the results of the averaged value of the predicted missile diameter over time and Figure 4.9 shows all the predictions made from based on the ANNs from each of the data sets.

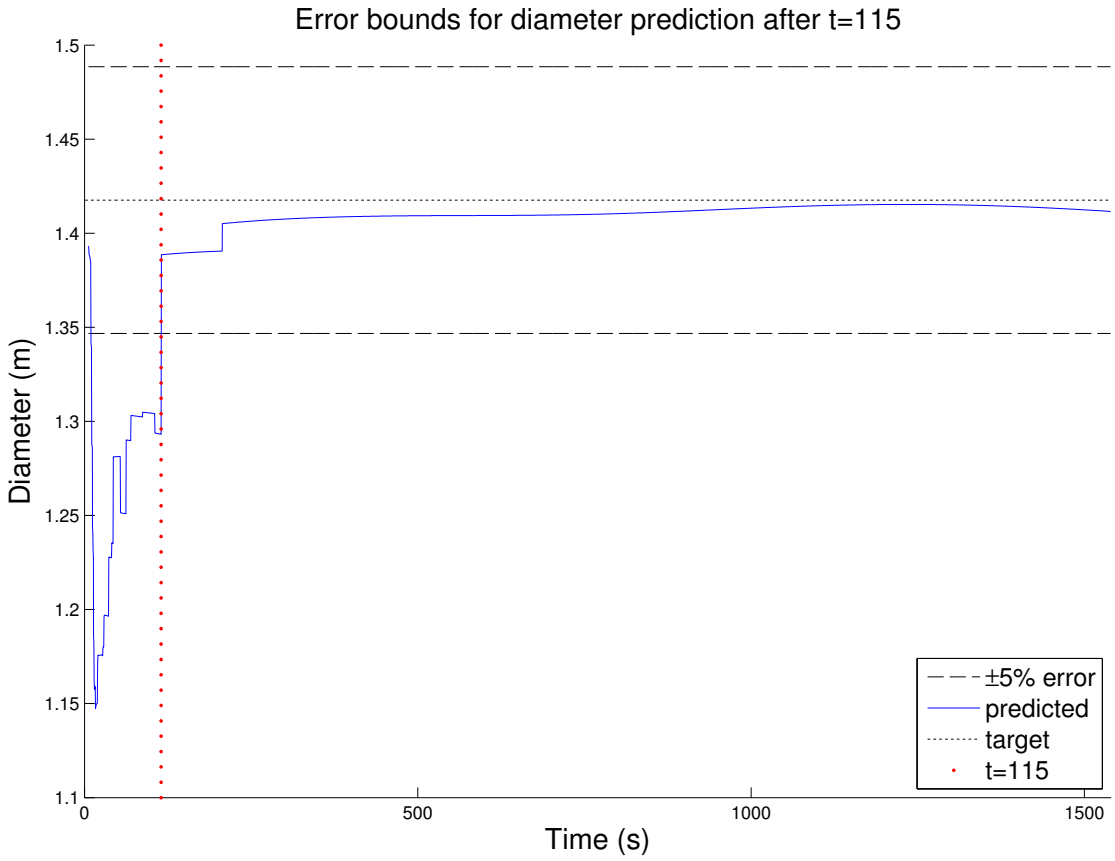


Figure 4.8: Averaged long-range missile prediction results

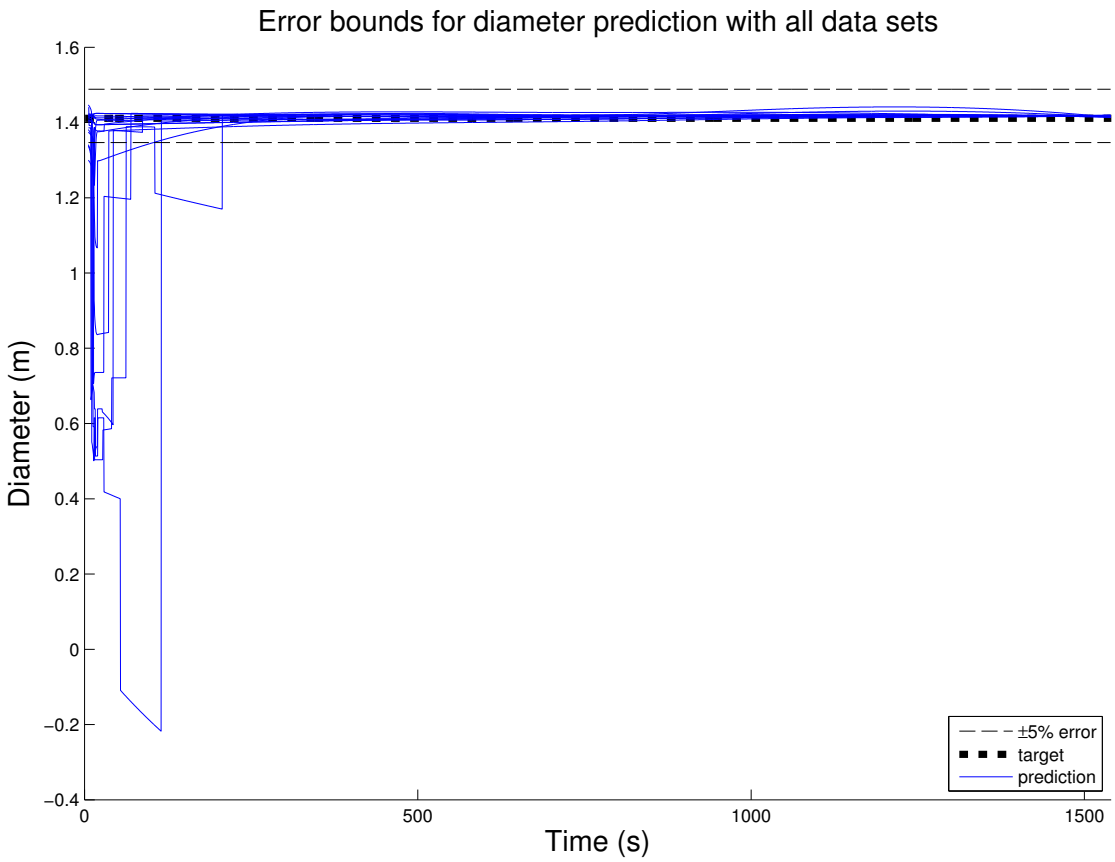


Figure 4.9: Long-range prediction results from all data sets

This set of plots show the predicted diameter over the course of a long-range trajectory, relative to the other missiles in the data set. The results obtained for this particular missile/trajectory pair show the viability of this method. One of the noticeable positive trends is that as more observations are processed in the characterization algorithm, the predicted diameter converges to the target actual diameter. Another aspect to note is the quick convergence; after 115 seconds after launch the missile is identified within the error limits and remains within the bounds. This convergence is not gradual, however, and in the first 115 seconds after launch the predictions made do not come close to the error bounds, which can clearly be seen in Figure 4.10.

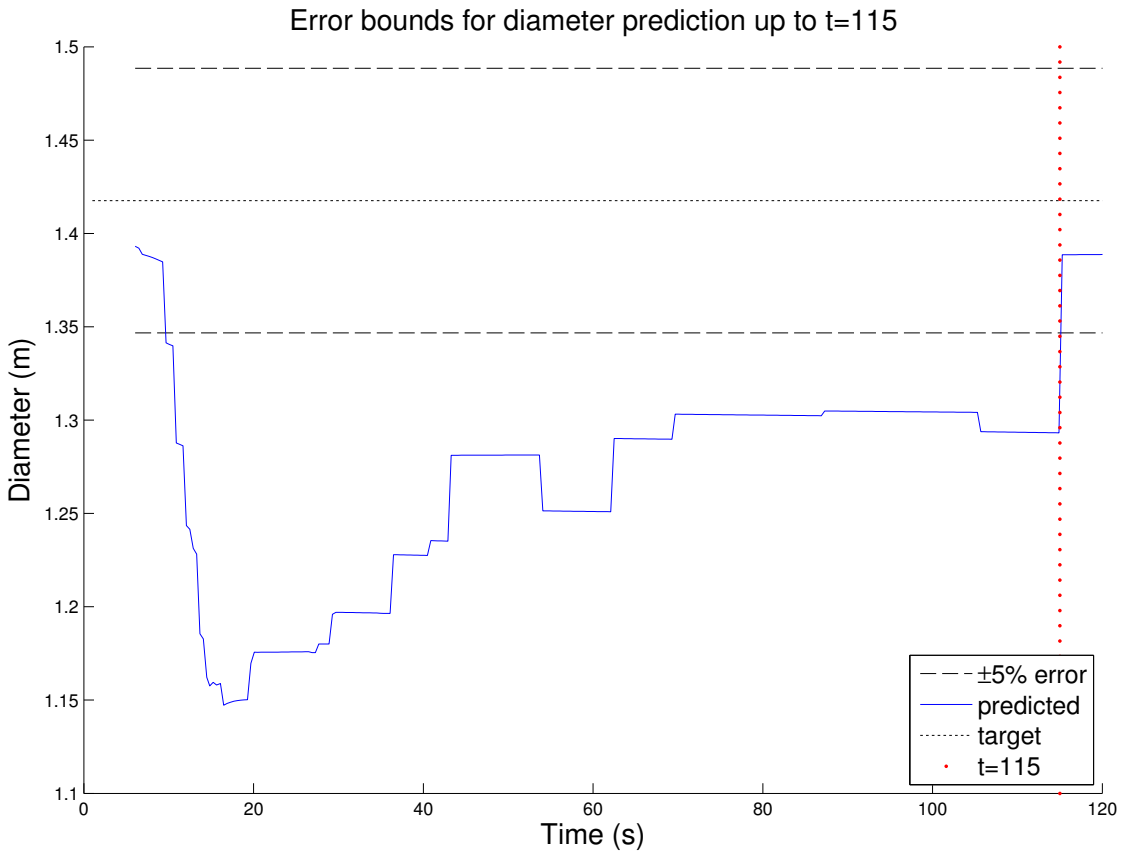


Figure 4.10: Averaged long-range missile prediction results up to 115 seconds

However, it is possible to predict the correct diameter before this time. By comparing the cluster assignment made during classification with that same missile's assigned cluster from the FFC, it was possible to determine at what times the classification algorithm correctly assigned cluster numbers. Some of the trials produced predictions that were accurate from the initial observations to the last. This was due to correct cluster assignments occurring 98% of the time. Table 4.1 shows on average how many times an improper cluster assignment was made in a single trial during the classification phase.

Stat	**Long-Range** total 3834 obs	**Short-Range** total 365 obs	**Mid-Range** total 1395 obs
Mean	76	267	1169
Std Dev	133	30	148
Percentage	2%	73%	85%

Table 4.1: Average inaccurate cluster assignments for each of the 3 missiles tested. The total observations are given for each test case.

Based on the results from the table, it would seem to follow that the short-range missile, with a success rate of 27%, would not manage to correctly predict target diameter within the acceptable error. While the short-range missile diameter is not predicted accurately as quickly as the long-range, accurate predictions are made after 110 seconds from launch, which can be seen from the averaged predictions in Figure 4.11 and in the full set of predictions in Figure 4.12.

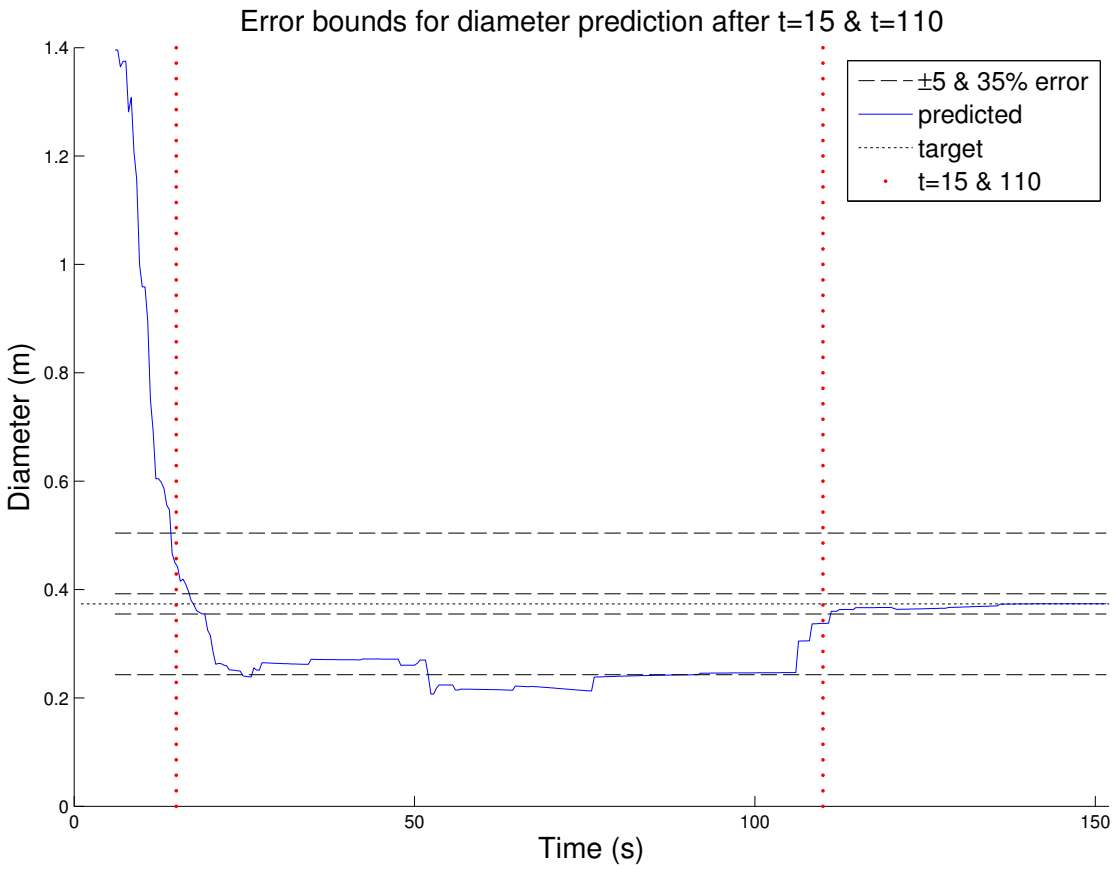


Figure 4.11: Short-range missile prediction results

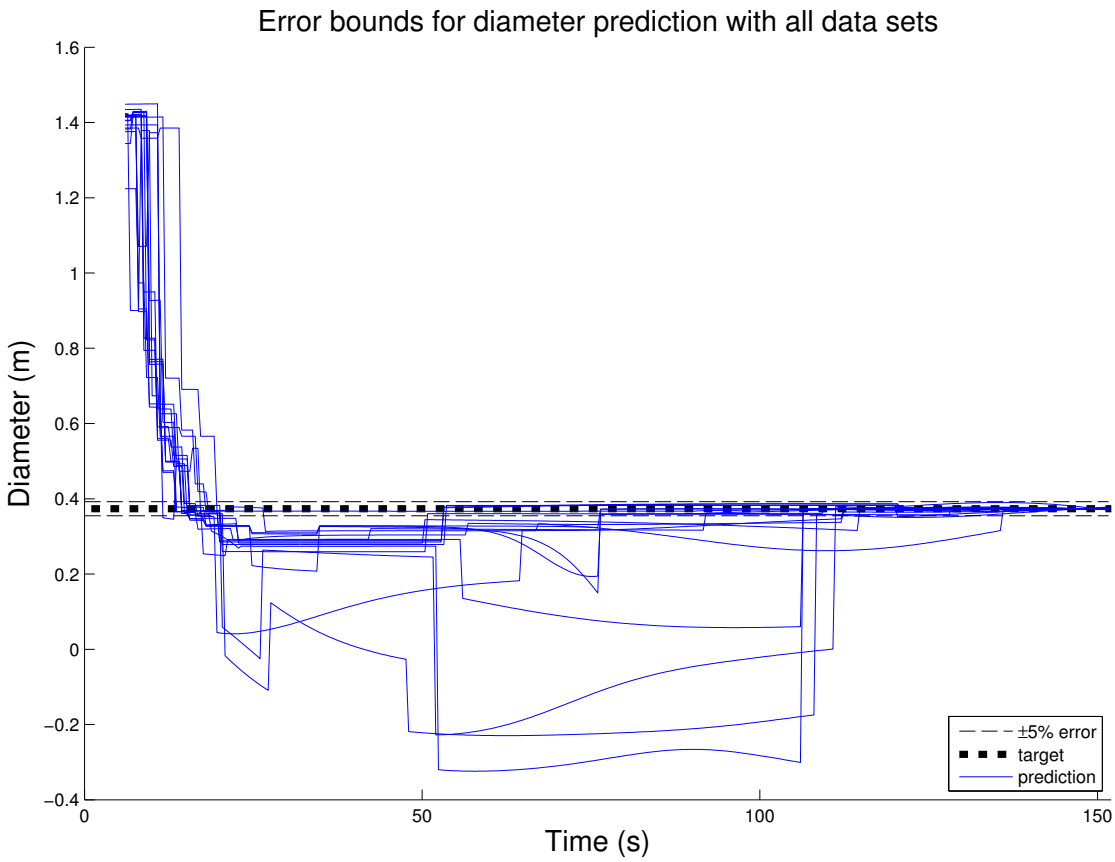


Figure 4.12: Short-range prediction results from all data sets

Most predictions do reach a steady-state early after 15 seconds, oscillating around the lower bound of 35%, but this error is outside acceptable limits. Some diameters are predicted as less than zero, an infeasible solution that results from having a bipolar sigmoid function. In future iterations of this algorithm, a binary sigmoid will be used. The worst predictions performed, however, were those during the characterization of the mid-range missile. These averaged results are shown in Figure 4.13 and all of the mid-range predictions are shown in Figure 4.14.

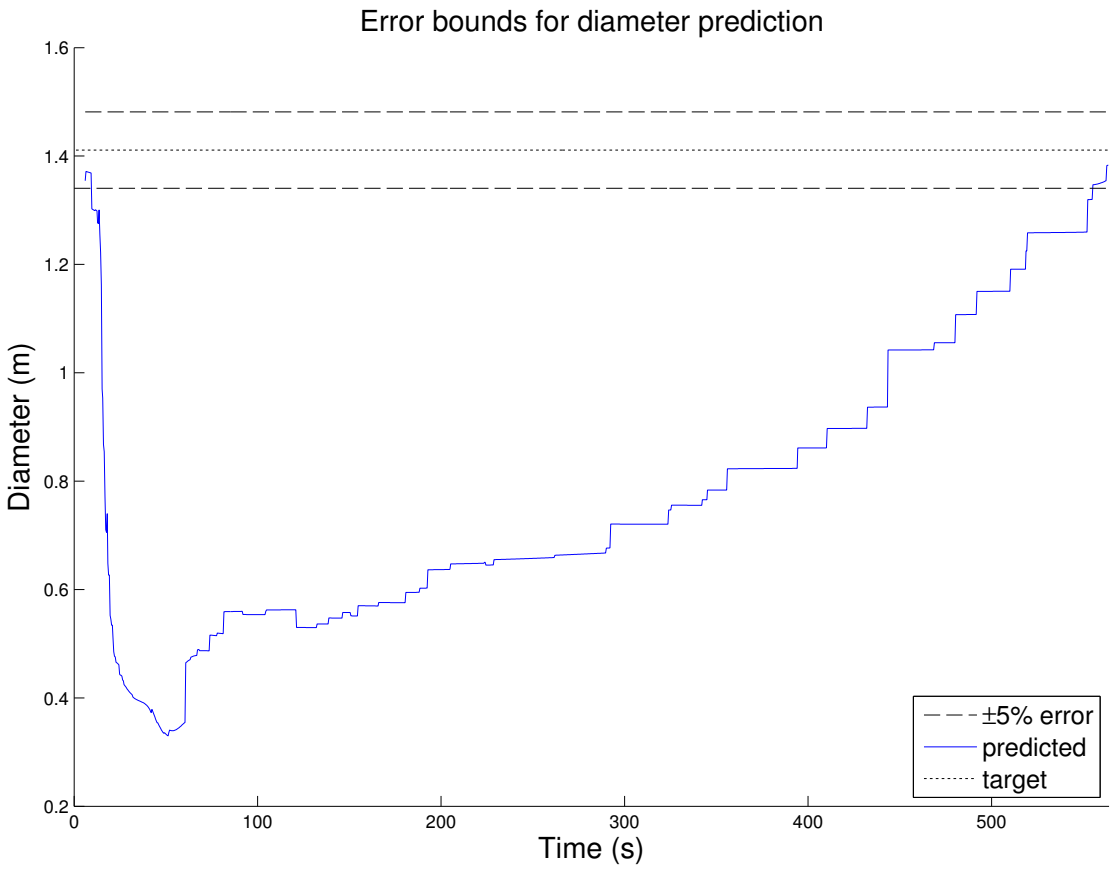


Figure 4.13: Mid-range missile prediction results

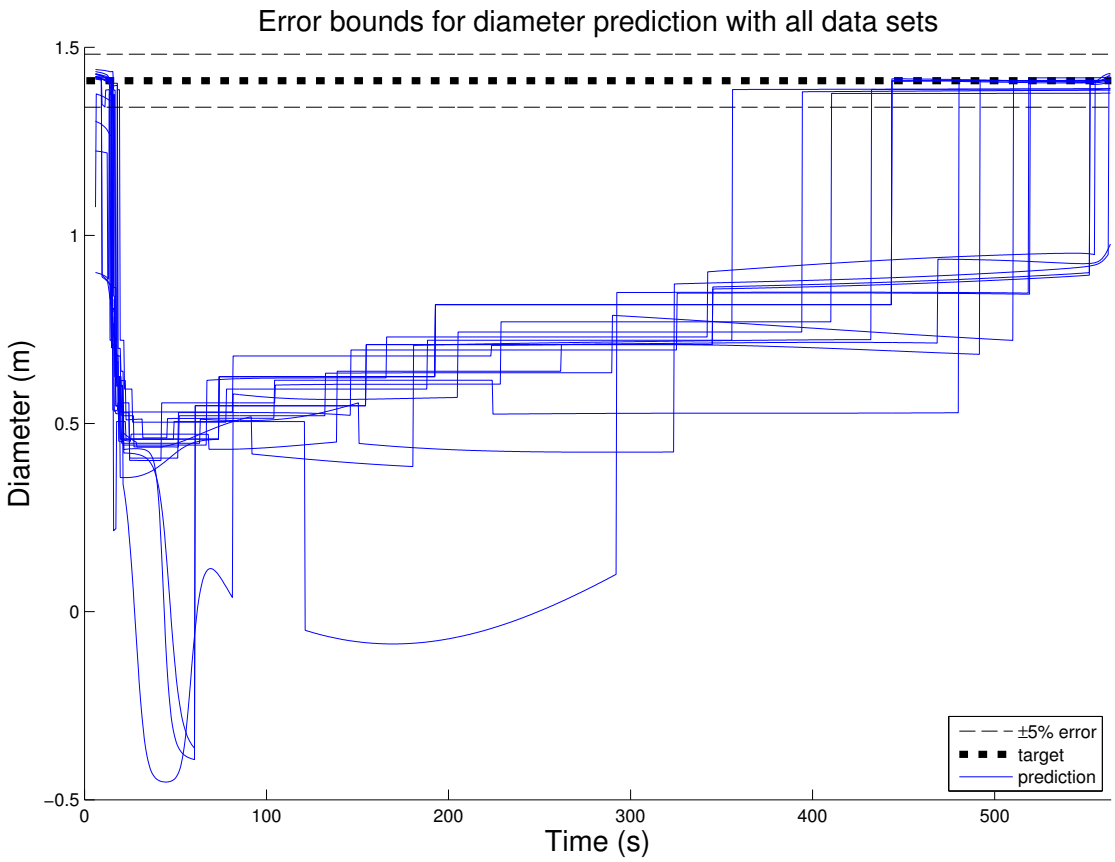


Figure 4.14: Mid-range prediction results from all data sets

Based on the information in Table 4.1, the mid-range missile was assigned to the correct cluster in only 15% of the observations tested. What is interesting about the prediction curve is that the initial predictions are within the 5% error limit. The predictions rapidly begin to become worse then slowly recover in the last tens of seconds. Furthermore, these predictions came from the same training sets as the long-range missile predictions which on average were within acceptable error.

One possible explanation as to why the discrepancy in the accuracy of the predictions across the difference missiles can be found in the size of the clusters each missile belonged to. Consistently, the long-range missile was placed in the second or third largest cluster of the data set, never the largest. The short-range missile was placed in the second or third smallest cluster, and the mid-range was placed in the largest cluster. The largest clusters in the twenty data sets were usually significantly larger than the next largest data set, sometimes 25-50%

larger. Due to this increased size, it is possible the large data sets did not produce a fully trained ANN. The same number of maximum iterations was implemented for all clustered data subsets; the large data sets might not have had enough iterations to minimize the error in the wieghts, leading to poor predictions for missiles in the largest cluster.

Another likely cause for the prediction accuracy inconsistency is the method of classification. Each of the tested missiles was able to be predicted within the acceptable error bounds at least once, meaning that a correct prediction was possible with the given trained ANN. Though improvements could be made to the training methods, an increase in the ANN prediction accuracy would not help if missiles are wrongly assigned to a cluster during classification. A mismatched assignment means that the missile's diameter will be predicted using the incorrect network. This mismatch occurs in the mid-range missiles particularly because most of their flight time falls within the range of some of the larger missiles' burn time. The figures presented in Section 4.2.1 show a tangled grouping of clusters for the first 10,000 feet; this creates a large difficulty in the ability to properly classify within this range. After this initial range, the high-velocity missiles (corresponding to long-range) are picked out easily since they lie within a more sparse region; however, the lower max velocity missiles lie within a more dense set of multiple clusters during the majority of their flight time. Though classification is easier than the initial moments after launch, the classification algorithm still has difficulty in proper classification, an assertion confirmed by the performance displayed in Figure 4.13.

Chapter 5

Conclusion

In summary, computational intelligence methods were used to to cluster, classify, and characterize missile systems in mid-flight. For this study, the diameter of the missile was chosen as the geometric characteristic to be predicted. Twenty data sets were generated using a missile flyout simulation program. These data sets were used as training sets for the FFC algorithm which produced 8 clusters for each respective data set. These subsets, coupled with the geometrical data, were used as the target data for the ANNs. Another ANN was trained without clustering prior to training in order to show the effects of the FFC. The results from this comparison showed that the clustering reduced the error in the neural network by an order of magnitude at the expense of creating and training multiple networks.

With the networks trained and the clusters found, simulated flyouts were performed for the same 3 missiles in every data set. The results were given as the predicted diameter over the time of the trajectory. It was found that the accuracy of the prediction for the missile's diameter seemed to be dependent on the ability of the classification algorithm to properly assign a cluster to the missile. In other words, a proper cluster assignment for the missile in question while in mid-flight would produce the proper diameter.

There are two important findings from this study: the use of a clustering method can decrease the overall training and validation error in an artificial neural network, and geometrical parameters of missiles systems can be predicted with the an ANN while in flight. While these results show a proof of concept, the overall accuracy and consistency of the algorithm can be improved, of which the most important is tuning the classification algorithm. It was shown in the results that more improper cluster assignments for a missile

will lead to poor predictions regardless of how well the suite of neural networks were trained. Moreover, it is vital that if this is to be used as an aid for missile defense applications that these cluster assignments not only are correct, but are correct in the first seconds of a launch. This issue was discussed in the results; most missile launches initially look similar so there is a need for a more discerning method of classification in these initial seconds. The fuzzy algorithm might still be suitable with the adjustment of the fuzzy parameter or perhaps the assignment method; neither the alpha cut method nor the method used produced consistent results.

In order to make a much more robust algorithm, various launch angles and fuel types will need to be added to the data sets. This will undoubtedly add much more complexity to the problem as this will decouple the relationship between position and diameter; the clustering and neural network training will be far more sensitive to the changed in velocity and possibly even acceleration. This would also mean finding another feature extraction method other than the conversion into RMS space using the reference missile.

Overall, the results show a promising continuation of aerospace applications using computational methods. With some improvements to the classification algorithm, the methodology could prove as a useful tool for missile defense or for other real-time characterization applications.

Bibliography

[1] Hartfield, R., Jenkins, R., and Burkhalter, J., "Ramjet Powered Missile Design Using a Genetic Algorithm," *AIAA Aerospace Sciences Meeting*, 2004.

[2] Jenkins, R., Hartfield, R., and Burkhalter, J., "Optimizing a Solid Rocket Motor Boosted Ramjet Powered Missile Using a Genetic Algorithm," *AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, 2005.

[3] Dyer, J., Hartfield, R., and Dozier, G., "Aerospace Design Optimization Using a Steady State Real-Coded Genetic Algorithm," *Applied Mathematics and Computation*, Vol. 218, No. 9, 2012, pp. 4710–4730.

[4] Albarado, K., Hartfield, R., Hurston, B., and Jenkins, R., "Solid Rocket Performance Matching Using Pattern Search/Particle Swarm Optimization," *International Journal of Aerospace Engineering*, 2012.

[5] Kiyak, Z., *Ant Colony Optimization: An Alternative Heuristic for Aerospace Design Applications*, Ph.D. thesis, Auburn University, December 2013.

[6] Sanders, G. A. and Washington, W. D., "Computer Program for Estimating Stability Derivatives of Missile Configurations - Users Manual," Tech. rep., U.S. Army Missile Command, 1982.

[7] Abraham, C., "Unsupervised Curve Clustering using B-Splines," *Scandanavian Journal of Statistics*, Vol. 30, No. 3, 2003, pp. 581–595.

[8] Luan, Y. and Li, H., "Clustering of time-course gene expression data using a mixed-effects model with B-splines," *Bioinformatics*, Vol. 19, No. 4, 2003, pp. 474–482.

[9] Tarpey, T. and Kinateder, K., "Clustering Functional Data," *Journal of Classification*, Vol. 20, No. 1, 2003, pp. 93–114.

[10] Ray, S. and Mallick, B., "Functional clustering by Bayesian wavelet methods," *J.R. Statist. Soc. B*, Vol. 68, No. 2, 2006, pp. 305–332.

[11] Chiou, J.-M. and Li, P.-L., "Functional clustering and identifying subtructures of longitudinal data," *J.R. Statist. Soc. B*, Vol. 69, No. 4, 2007, pp. 679–699.

[12] Lei, J. and Lu, C., "Target classification based on micro-Doppler signatures," *IEEE International*, 2005, pp. 179–183.

[13] Albarado, K., Hartfield, R., Carpenter, D., Burkahlter, J., and Ritz, S., "Rapid Missile Classification for Early Launch Using Neural Networks," *MDissile Defence Conference*, 2012.

[14] Zadeh, L. A., "Fuzzy Sets," *Information and Control*, Vol. 8, No. 3, 1965, pp. 338–353.

[15] Ross, T. J., *Fuzzy Logic with Engineering Applications*, John Wiley & Sons, West Sussex, UK, 3rd ed., 2010.

[16] MacQueen, J. B., "Some Methods for Classification and Analysis of Multivariate Observations," *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[17] Bezdeck, J. C., Ehrlich, R., and Full, W., "FCM: The Fuzzy c-Means Clustering Algorithm," *Computers and Geosciences*, Vol. 10, No. 2-3, 1984, pp. 191–203.

[18] Fausett, L., *Fundamentals of Neural Networks: Architectures, Algorithms, and Applications*, Prentice-Hall, Inc., Upper Saddle River, NJ, 1st ed., 1994.

[19] McCulloch, W. and Pitts, W., "A Logical Calculus of the Ideas Immantent in Nervous Activity," *Bulletin of Mathemetical Biophysics*, Vol. 5, No. 1, 1943, pp. 115–133.

[20] Minsky, M. and Papert, S., *Perceptrons: An Introduction to Computational Geometry*, MIT Press, Cambridge, MA, 1st ed., 1969.

[21] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning representations by back-propagating errors," *Nature*, Vol. 323, No. 10, 1986, pp. 533–536.

[22] Hopfield, J. J., "Neural networks and physical systems with emergent collective computational abilities," *Proceeding of the National Academy of Sciences of the United States of America*, Vol. 79, No. 8, 1982, pp. 2554–2558.

[23] Duda, R., Hart, P., and Stork, D., *Pattern Classification*, John Wiley and Sons, Inc., New York, NY, 2nd ed., 2001.

[24] Navidi, W., *Statistics for Scientist and Engineers*, McGraw-Hill Companies, Inc., New York, NY, 3rd ed., 2011.

Appendix A

Values for SRM code parameters

4.18E-01	1	rnose/rbody FIN CNTL	1.35E+00	19	tail semispan/dbody
1.72E+00	2	lnose/dbody	1.19E+00	20	tail root chord = crt/dbody
5.00E+00	3	fuel type	7.93E-01	21	tail taper ratio = ctt/crt
5.62E-01	4	star out R rpvar=(rp+f)/rbody1	7.17E+00	22	LE sweep angle deg
1.70E-01	5	star inner ratio=ri/rp	9.92E-01	23	xTEt xTEt/lbody
4.61E+00	6	number of star pts	5.00E+03	24	auto pilot delay time sec
9.70E-02	7	fillet radius ratio=f/rp	6.93E+01	25	initial launch angle deg
8.72E-01	8	eps (star PI*eps/N) width	3.98E+00	26	pitch multiplier gain
9.56E+00	9	star point angle deg	1.66E+00	27	yaw multiplier gain
7.90E-01	10	fractional noz len f/ro	1.41E+00	28	dumy
2.64E-01	11	Dia throat/Dbody=Dstar/Dbody	3.16E+00	29	initial pitch command angle - degrees
1.30E+01	12	Fineness ratio Lbody/Dbody	2.47E-02	30	initial yaw command angle - degrees
1.41E+00	13	dia of stage1 meters	5.22E-01	31	b2var=b2vane/rexit
1.13E-03	14	wing semispan/dbody	3.55E-01	32	time step to actuate fins (sec)
1.29E-03	15	wing root chord = crw/dbody	9.11E-04	33	correction to initial psi ang (deg)
9.28E-01	16	taper ratio = ctw/crw	4.00E+04	34	deltx for z corrections
1.23E+00	17	wing LE sweep angle deg	4.00E+04	35	deltx for y corrections
3.48E-01	18	xLE xLEw/lbody			

Appendix B

FFC Algorithm

```
% Author: Steven Ritz
% This program is part of the fuzzy neural suite of codes to sort, cluster,
% and predict missile geometery based on data that would be acquired
% through radar.
%
%   Fuzzy/ANN Suite:
%       -DataManager.m
%       -FuzzyClusterRobust.m*
%       -Max_Alt_Range.m
%       -plotting_things.m
%       -PreNeural.m
%       -RefMissileSort.m
%       -RealtimeFuzzyClassification.m
% NOTE: This requires the database 'Classificated_plus.mat' from
%       DataManager.m to be in current directory
% clc
% clear all
% close all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This is the function version of the code to be used for confidence
% interval testing. It will likely be put into the final suite version to
% be implemented into the GUI.
function [F]=FuzzyClusterRobust(m,c,epochs,allRef,refm,database_num)
close all
current_dir=strcat(pwd,sprintf('\\DataSet%i\\',database_num));
```

```
load(strcat(current_dir,'Classificated_plus.mat'));
%% Method
% Normalize
% Randomly Assign to clusters
% -> Compute distance from all obs to cluster center
% |  update membership values
% -< repeat until desired epochs is met
%% Constants
switch allRef
    case 0
        cycledM=refm;
    case 1
        cycledM=[1:1:total];
end
%% File Management
number_for_file=num2str(c);
file_path=strcat(current_dir,'\',number_for_file,' Clusters\');
dircheck=isdir(file_path);
if dircheck==0
    mkdir(file_path);
    mkdir(strcat(file_path,'\Trajectories\'));
    mkdir(strcat(file_path,'\Missiles In Clusters\'));
    mkdir(strcat(file_path,'\RMS Plots\'));
    mkdir(strcat(file_path,'\RMS Centers\'));
    mkdir(strcat(file_path,'\Ref Based DBs\'));
end
for refmissile=cycledM
    close all
%% Initialize Membership for all obs
for C=1:c
    eval(sprintf('Class%i.membership=zeros(1,total);',C));
end
for i=1:total
```

```
    C=rem(i,c);
    if C==0
        C=c;
    end
    eval(sprintf('Class%i.membership(i)=1;',C));
end
%% Reference Frame Calculation
eval(sprintf('ref.alt=spline(missile.m%i.time,missile.m%i.Alt);',...
    refmissile,refmissile));
eval(sprintf('ref.range=spline(missile.m%i.time,missile.m%i.Range);',...
    refmissile,refmissile));
eval(sprintf('ref.vel=spline(missile.m%i.time,missile.m%i.Velocity);',...
    refmissile,refmissile));
%% Coordinate Calculation
% Cycle through missiles, i
for i=1:total
    Coord.sosAlt=0;
    Coord.sosRange=0;
    Coord.sosVel=0;

    eval(sprintf('time=missile.m%i.time;', i));
    eval(sprintf('Alt=missile.m%i.Alt;', i));
    eval(sprintf('Range=missile.m%i.Range;', i));
    eval(sprintf('Vel=missile.m%i.Velocity;', i));
    for t=1:length(time)
            Coord.sosAlt=(ppval(ref.alt,time(t))-Alt(t))^2+Coord.sosAlt;
            Coord.sosRange=(ppval(ref.range,time(t))-Range(t))^2+...
                Coord.sosRange;
            Coord.sosVel=(ppval(ref.vel,time(t))-Vel(t))^2+Coord.sosVel;
    end
        Coord.RMSAlt(i)=sqrt(Coord.sosAlt)/length(time);
        Coord.RMSRange(i)=sqrt(Coord.sosRange)/length(time);
        Coord.RMSVel(i)=sqrt(Coord.sosVel)/length(time);
```

```
end
for ecount=1:epochs
%% Repeated Center Calculation
for C=1:c
    x1t=0; x2t=0; x3t=0; x1b=0; x2b=0; x3b=0;
    for i=1:total
        eval(sprintf('mum=Class%i.membership(i);',C));
        x1t=x1t+(mum^m*Coord.RMSAlt(i));
        x1b=x1b+(mum^m);
        x2t=x2t+(mum^m*Coord.RMSRange(i));
        x2b=x2b+(mum^m);
        x3t=x3t+(mum^m*Coord.RMSVel(i));
        x3b=x3b+(mum^m);

    end
    x1=x1t/x1b;
    x2=x2t/x2b;
    x3=x3t/x3b;
    eval(sprintf('Coord.Class%i=[x1,x2,x3];',C));
end
%% "Distance" Calculations
% Cycle through missiles, i
for i=1:total
    for C=1:c
        eval(sprintf('Class%i.sosX1=0;',C));
        eval(sprintf('Class%i.sosX2=0;',C));
        eval(sprintf('Class%i.sosX3=0;',C));
    end
    eval(sprintf('time=missile.m%i.time;', i));
    eval(sprintf('Alt=missile.m%i.Alt;', i));
    eval(sprintf('Range=missile.m%i.Range;', i));
    eval(sprintf('Vel=missile.m%i.Velocity;', i));
    for C=1:c
```

```
        eval(sprintf('Altdiff=Coord.Class%i(1)-Coord.RMSAlt(i);',C));
        eval(sprintf('Rangediff=Coord.Class%i(2)-Coord.RMSRange(i);',C));
        eval(sprintf('Veldiff=Coord.Class%i(3)-Coord.RMSVel(i);',C));
        eval(sprintf(...
            'Class%i.Distance(i)=sqrt(Altdiff^2+Rangediff^2+Veldiff^2);',...
            C));
    end
end
%% Membership Values
for C=1:c
    eval(sprintf('Class%i.members=[];',C));
end
for i=1:total
    for C=1:c
        eval(sprintf('Class%i.membership(i)=0;',C));
        for CC=1:c
            eval(sprintf(...
'Class%i.membership(i)=Class%i.membership(i)+...
    (Class%i.Distance(i)/Class%i.Distance(i))^(2/(m-1));',C,C,C,CC));
        end
        eval(sprintf(...
            'Class%i.membership(i)=Class%i.membership(i)^(-1);',C,C));
    end
    member_check=Class1.membership(i);
    cluster_id=1;
    for C=2:c
        if eval(sprintf('Class%i.membership(i)>member_check',C))
            eval(sprintf('member_check=Class%i.membership(i);',C));
            cluster_id=C;
        end
    end
    j=length(eval(sprintf('Class%i.members;',cluster_id)));
     eval(sprintf('Class%i.members(j+1)= i;',cluster_id));
```

```
end
%% Cleanup
for C=1:c
     eval(sprintf('epoch_center{C,ecount}=Coord.Class%i;',C));
end
end
%% Partition Coef
% F=0;
% for i=1:total
%     for C=1:c
%         eval(sprintf('F=F+Class%i.membership(i)^2/total;',C));
%     end
% end
%% Validation by plot
altlim=0;rangelim=0;vellim=0;
for t=1:epochs
    for i=1:c
        centerplot=epoch_center{i,t};
        if centerplot(1)>altlim
            altlim=centerplot(1);
        end
        if centerplot(2)>rangelim
            rangelim=centerplot(2);
        end
        if centerplot(3)>vellim
            vellim=centerplot(3);
        end
    end
end
for C=1:c
    eval(sprintf('allClasses.Class%i=Class%i;',C,C));
end
[fig_handel,plot_handel]=...
```

```
    plotting_things(c,allClasses,Coord,refmissile,file_path);
plotting_things_str=strcat(pwd,'\plotting_things.m');
close all
% %  Plot Things
colours={'-m' '-r' '-g' '-b' '-k' '-xr' '-xm' '-xg' '-xb' '-xk'};
fig_handel=figure('OuterPosition',[200 400 850 800]);
plot_handel=gca;
for C=1:c
    if eval(sprintf('sum(Class%i.members)',C))>0
        for i=1:eval(sprintf('length(Class%i.members)',C))
            pp=1;
            velplot=0;
            Rangeplot=0;
            Altplot=0;
            eval(sprintf('j=Class%i.members(i);',C));
            eval(sprintf('time=missile.m%i.time;',j));
            for plotter=1:length(time)
                if mod(plotter,10)==0
                eval(sprintf(...
                    'velplot(pp)=missile.m%i.Velocity(plotter);',j));
                eval(sprintf(...
                    'Rangeplot(pp)=missile.m%i.Range(plotter);',j));
                eval(sprintf(...
                    'Altplot(pp)=missile.m%i.Alt(plotter);',j));
                pp=pp+1;
                else
                end
            end
            eval(sprintf(...
'plot_handel,plot(Rangeplot,velplot,''%s'',''MarkerSize'',4);',colours{C}));
            hold(plot_handel,'on');
        end
    end
```

```
end
xlabel('Range','FontSize',14);ylabel('Velocity','FontSize',14);
xlim([0 100])
export_fig(strcat(file_path,...
    '\Trajectories\',sprintf('%icluster_%i.pdf',c,refmissile)),...
    '-transparent')

Membership_file=fopen(strcat(file_path,...
    '\Missiles In Clusters\',sprintf('RefMissile%i.txt',refmissile)),'w');
for i=1:c
    eval(sprintf('allmembers(i)=length(Class%i.members);',i));
    eval(sprintf('thememberss=Class%i.members;',i));
    fprintf(Membership_file,'Class %i \r\n',i);
    for b=1:allmembers(i)
        fprintf(Membership_file,'%i  ',thememberss(b));
    end
    fprintf(Membership_file,' \r\n');

end
F=std(allmembers);
fclose(Membership_file);
save(strcat(file_path,'\Ref Based DBs\',...
    sprintf('Clustered_ref_%i.mat',refmissile)));
end
end
```