2006

Fuzzy-Neural Cost Estimation for Engine Tests

Edit J. Kaminsky
University of New Orleans, ejbourge@uno.edu

Holly Danker-McDermott
University of New Orleans

Freddie Douglas

Recommended Citation

Kaminsky, E., H. Danker-McDermott, and F. Douglas, "Fuzzy-Neural Cost Estimation for Engine Tests", Chapter 9 in Computational Economics: A Perspective from Computational Intelligence, Idea Group Publishing (Hershey, PA), ISBN: 1-59140-649-8, 2006, pp. 178-204.

Computational Economics:
A Perspective from Computational Intelligence

Shu-Heng Chen
National Chengchi University, Taipei, Taiwan

Lakhmi Jain
University of South Australia, Adelaide, Australia

Chung-Ching Tai
National Chengchi University, Taipei, Taiwan

IDEA GROUP PUBLISHING
Hershey • London • Melbourne • Singapore

Acquisitions Editor: Michelle Potter
Development Editor: Kristin Roth
Senior Managing Editor: Amanda Appicello
Managing Editor: Jennifer Neidig
Copy Editor: Becky Shore
Typesetter: Diane Huskinson
Cover Design: Lisa Tosheff
Printed at: Integrated Book Technology

Published in the United States of America by
Idea Group Publishing (an imprint of Idea Group Inc.)
701 E. Chocolate Avenue, Suite 200
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@idea-group.com
Web site: http://www.idea-group.com

and in the United Kingdom by
Idea Group Publishing (an imprint of Idea Group Inc.)
3 Henrietta Street
Covent Garden
London WC2E 8LU
Tel: 44 20 7240 0856
Fax: 44 20 7379 3313
Web site: http://www.eurospan.co.uk

Library of Congress Cataloging-in-Publication Data

Computational economics : a perspective from computational intelligence / Shu-Heng chen and Lakhmi Jain, editors.
p. cm.
Summary: "This book identifies the economic as well as financial problems that may be solved efficiently with computational methods and explains why those problems should best be solved with computational methods"--Provided by publisher.
Includes bibliographical references and index.
ISBN 1-59140-649-8 (hardcover) -- ISBN 1-59140-650-1 (softcover) -- ISBN 1-59140-651-X (ebook)
1. Economics--Data processing. 2. Economics, Mathematical. 3. Finance--Data processing. I. Chen, Shu-Heng, 1959- II. Jain, L. C.
HB143.5.C663 2006
330'.0285--dc22
2005020633

Computational Economics: A Perspective from Computational Intelligence is part of the Idea Group Publishing series named *Computational Intelligence and Its Applications Series*.

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter IX

Fuzzy-Neural Cost Estimation for Engine Tests

Edit J. Kaminsky
University of New Orleans, USA

Holly Danker-McDermot
New Orleans, USA

Freddie Douglas, III
NASA, Stennis Space Center, USA

ABSTRACT

This chapter discusses artificial computational intelligence methods as applied to cost prediction. We present the development of a suite of hybrid fuzzy-neural systems for predicting the cost of performing engine tests at NASA's Stennis Space Center testing facilities. The system is composed of several adaptive network-based fuzzy inference systems (ANFIS), with or without neural subsystems. The output produced by each system in the suite is a rough order of magnitude (ROM) cost estimate for performing the engine test. Basic systems predict cost based solely on raw test data, whereas others use preprocessing of these data, such as principal components and locally linear embedding (LLE), before entering the fuzzy engines. Backpropagation neural networks and radial basis functions networks (RBFNs) are also used to aid in the cost prediction by merging the costs estimated by several ANFIS into a final cost estimate.

INTRODUCTION

John C. Stennis Space Center (SSC) is NASA's primary center for testing and flight certification of rocket propulsion systems for the space shuttle and future generations of space vehicles. Because of its important role in engine testing for more than 3 decades, SSC has been designated NASA's Center of Excellence for Rocket Propulsion Testing. SSC tests all space shuttle main engines (SSME). These high-performance, liquid-fueled engines provide most of the total impulse needed during the shuttle's 8 1/2-minute flight into orbit. All SSME must pass a series of test firings at SSC prior to being installed in the back of the orbiter. Moreover, commercial engine and component tests are also performed at the SSC NASA facilities.

A few operations management software systems, including cost estimating algorithms, have been developed in the past (Lockheed Martin Space Operations, 2001; Lockheed Martin Space Operations, 2000; Rocket Propulsion Testing Lead Center, 1997, 1998; Sundar, 2001) to aid in scheduling and managing tests as well as to predict the cost of performing component and engine tests at NASA's John C. Stennis Space Center testing facilities: The cost estimating model (CEM), which includes cost estimating relationships (CER), the operations impact assessor (OIA), bottoms-up cost estimator (BUCE), and risk constrained optimized strategic planning (RCOSP). The results, however, have not been very encouraging and are not available in the open literature. OIA and RCOSP are very complex systems and require input data that are rarely, if ever, available before tests are performed. BUCE is a bottoms-up estimator and requires a level of detail for the input data (e.g., a complete list of parts and number of labor hours) that bans this tool from being used to generate a rough order of magnitude estimate. CEM is the simplest system and it prompts the user to input the same type of preliminary data as the systems presented in this Chapter. Results from CEM will be compared to the new computational intelligence systems which perform considerably better. CEM uses cost estimating relationships, parametric estimation, and statistics.

In this chapter, we present a system for this same purpose (cost prediction), based on adaptive network-based fuzzy inference systems (ANFIS) and neural networks (NN). The hybrid software suite was developed in Matlab[1] and combines the adaptive capabilities of neural networks and the ease of development and additional benefits of fuzzy logic based systems, detailed by the current authors in (Danker-McDermot, 2004; Kaminsky, 2002; Kaminsky & Douglas, 2003). The software-based system consists of several user-selectable subsystems ranging from simple fuzzy estimators, to medium complexity ANFIS systems that use normalized and transformed input data as well as more complex multistage fuzzy-neural or neural systems. We will discuss each here, and present comparative results indicating that these artificial intelligence procedures produce good cost estimates even when they are developed using very small sets of data. The accuracy of the predicted cost increases as the complexity of the system (as measured by number of processing routines, number of stages, and number of input variables) increases.

The goal of the project[2] was to develop a hybrid fuzzy-neural cost estimating system to obtain rough order of magnitude (ROM) estimates of the cost for both component and engine tests. A very small set of data, mainly from NASA's Project Requirement Documents (PRD) (NASA, 2001; University of New Orleans, 2000), were available for component and engine tests performed at NASA's John C. Stennis Space Center (SSC).

In this chapter, however, we detail only the hardest problem: predicting cost for engine tests. The available PRD data set for engine tests was much smaller and more incomplete than the component test sets. Results presented here are, therefore, easy to improve upon for component tests. For results of component tests, the reader may refer to Kaminsky (2002) and Kaminsky and Douglas (2003). A subset of the already small group of PRD data for engine tests was used to train the computational intelligence fuzzy-neural systems in the suite. The trained systems are then used to predict cost for unseen engine articles (the testing set).

Several prototypes were developed and are described in the rest of this chapter: Simple ANFIS cost estimators (ANFIS), principal component analysis (PCA) ANFIS cost estimators (PCA-ANFIS), parallel/cascaded ANFIS systems (Parallel-ANFIS), locally linear embedding (LLE) ANFIS estimators (LLE-ANFIS), fuzzy-neural estimators (Parallel-ANFIS-NN), and radial basis function network estimators (RBFN). These differ in complexity and amount of preprocessing needed. Accuracy of predicted cost, although similar in order of magnitude, varies depending on the complexity of the system.

Principal components and LLE are used as preprocessing stages to reduce the dimensionality of the data because we have many more variables (descriptors) than we have exemplars (articles in the training set). PCA yields a linear decomposition whereas LLE is a nonlinear reduction method.

The rest of this chapter is organized as follows: In the next section, the engine test data, data analysis, and the preprocessing routines used are presented. We then briefly summarize ANFIS theory and present the various prototypes, followed by results for each of these prototypes and comparative results among the various systems. A summary and suggestions for further work are given along with conclusions.

DATA DESCRIPTION AND PREPROCESSING

This section discusses the data, collected and provided by NASA at Stennis Space Center, used to develop and test the fuzzy-neuro systems. We first describe the raw data and their limitations, and later analyze these data. We also discuss preprocessing of the raw data.

Data Description

The systems developed are supervised (i.e., they are developed using training data to produce the mapping sought). The nonlinear mapping is from raw input data to output cost. The raw data, then, are of extreme importance, both in quality and in quantity. As many project requirements descriptions (PRDs; NASA, 2001) as possible were collected. These PRDs characterize the engines tested at SSC. Unfortunately, the total number of articles is very small, generating small sets of training and testing data. A total of only 11 articles are complete enough to be used. These data have been used in two ways: to develop the models using a training subset and to test the newly developed models with a testing set previously unseen by the trained systems. The cost of performing the tests for these articles ranged from a few hundred thousand dollars to about 12 million dollars.

PRDs contained 18 variables that had data for at least one article. Many of these variables, however, had no data for most of the articles, and had to be discarded or filled by methods discussed later in this Chapter. The PRD input data variables left are given in Table 1. All variables in this table were at least considered for use, but some were

sometimes discarded after analysis of the predictive value of the variable indicated that they were of little use in predicting the cost of performing tests for the particular system under consideration. Not all variables were used in all prototyped systems.

A particular engine article has the data given in the last column of Table 1. Notice that variables 4-6 are codes (integer numbers) indicating the type of fuel, pressurant, and oxidizer. Codes 4, 6 and 8 are used, respectively, for GHe (gaseous helium), H_20_2 (hydrogen peroxide), and JP8 (jet propulsion fuel type 8). Test stand code 3 indicates NASA's stand E3 at Stennis Space Center. Data is not available for this article for variables 13-15. The cost to perform this test was slightly over $700,000.

The already extremely small collection of 11 sets of engine article data was randomly separated into training and testing sets. For some of our systems we used 6 articles for training and 5 for testing, while for others we increased the training set to 7 and reduced the testing set to 4 articles. The articles in the testing sets are only used to test the generalization ability of the cost prediction systems and were not used at all in the development of the ANFIS or the NNs.

Data Analysis

The number of data variables available (18) was larger than the total number of articles (11). When dealing with fuzzy systems or neural networks, it is always preferable to have more vectors in the set than the number of elements in those vectors. This was a large problem for the NASA engine test data because there were only 11 viable data exemplars, each with a maximum dimensionality of 19 when cost is included as the last variable. We need to somehow reduce the dimensionality of the set but must ensure that we do not discard any of the most important (most predictive) variables. In order to determine which data variables to discard, the information within and predictive value of the various variables had to be analyzed. Exhaustive and sequential searches were performed to determine the input attributes that have the most prediction power for ANFIS modeling. The exhaustive search, by its nature, yields the best results; however, it is extremely time consuming and computationally expensive.

In summary, variables 1, 2, 4, 7, 13 and 17 in Table 1 were the only ones that repeatedly showed to have predictive power for engine tests using the exhaustive search. When we used the sequential search mechanism, similar conclusions were reached,

Table 1. Input variables for engine tests (from PRDs)

No.	*Name*	*Description*	*Example Data*
1	DuratDd	Duration of test in days	45 days
2	NoTest	Number of tests	25 tests
3	TestDurMax	Maximum duration of test	200 sec
4	Fuel	Fuel code (integer)	8
5	Pressurant	Pressurant code (integer)	4
6	Oxidizer	Oxidizer code (integer)	6
7	Thrust	Thrust	5 450 lbs
12	ThrustMeas	Thrust Measurement (Boolean)	0
13	FuelFlow	Rate of fuel flow	N/A
14	PressuraPr	Pressure of pressurant	N/A
15	OxidizerFl	Rate of oxidizer flow	N/A
17	TestStand	Test stand code (integer)	3
19	TotalCost	Total cost of performing test	$702 000

except that variables 3, 5 and 6 also showed to be important in a few cases. The output variable in all cases is the total cost of the test (variable 19).

As an example, we summarize the relative importance of the three most important variables as a function of the number of fuzzy membership functions (from 2 to 4) in Table 2. Clearly the predictive power of a given variable depends on the number of membership functions allowed for that particular variable. Thrust rarely appeared as the most predictive variable, but it appeared as a variable to be included in almost all runs.

Missing Data

Another problem with the engine test data at our disposal is that frequently there is information missing for an article, but which piece of information was missing changed with each article. Ideally, if the information cannot be found for all articles, it would be best to eliminate these variables entirely. This is not a viable option in our situation, however, because almost all of the data variables have their value missing for at least one article. There does not seem to be a large body of published research dealing with small and incomplete data sets. The work we did find dealt mainly with incomplete data sets in neural classification systems. Ishibuchi, Miyazaki and Tanaka (1994) proposed a method for dealing with incomplete data by using an interval representation of incomplete data with missing inputs. After a network is trained using learning algorithms for interval training data, a new sample consisting of the missing inputs is presented along with an interval vector. The output from the neural network is also an interval vector. This output is then classified using four definitions of inequality between intervals.

Granger, Rubin, Gorssberg and Lavoie (2000) proposed using a fuzzy ARTMAP neural network to deal with incomplete data for a classification problem. This approach presented the fuzzy ARTMAP with an indicator vector that described whether a data component was present or not. Unlike replacement methods, the weight vector is modified as well as the input vector in response to missing components.

Another method to deal with incomplete data is to use the normal information diffusion model, which divides an observation into many parts according to a normal function (Chongfu, 1998). This technique attempts to find a suitable membership function to represent a fuzzy group that represents the incomplete data. This fuzzy group is then used to derive more data samples. Unfortunately, this method can be computationally intensive.

Finally, some other methods viable for the engine data test sets are the much simpler mean and multiple imputation. Mean imputation simply replaces the missing data with the mean value of the samples. This method can cause misleading results because the changed data cannot reflect the uncertainty caused by the missing data. Multiple imputation is similar to mean imputation, but the missing data are replaced by a set of

Table 2. Relative importance of the most important variables for engine tests with the number of ANFIS membership functions as a parameter

No. of MF	*Variables in order of importance* 1st	2nd	3rd
4	DuratDd	Fuel	Thrust
3	NoTests	Oxidizer	Thrust
2	DuratDd	Fuel	Thrust

possible values from their predictive distribution. This set reflects the uncertainty of the values predicted from the observed ones (Zhou, 2000). This method yields much better results than mean imputation, but it can also become computationally intensive.

In the work described here we use mean imputation, mode imputation, and median imputation. Mode imputation, where the most common value is used to fill in missing data, was used when codes (such as for fuel, pressurant, oxidizer, or test stand) were unknown. Mean imputation was used for FuelFlow, and median imputation (i.e., filling missing data with the median value of that variable over all training articles) was used to replace unknown values of the pressure of the pressurant, PressuraPr.

Dimensionality Reduction

Neural and fuzzy system training is performed more efficiently after certain processing routines are applied to the raw input data. Some of these processing routines, such as principal component analysis (PCA), not only expedite training, but also reduce the dimensionality of the data set and provide information about the data which is not obvious in their original state. Raw data were used in many cases, whereas in other cases preprocessing techniques were applied to the raw data for normalization, data transformation, and dimensionality reduction. We use the following preprocessing algorithms:

- Normalization to standard deviation of one and mean of zero
- Normalization to range of [-1,1]
- Principal components analysis (PCA)
- Locally linear embedding (LLE)
- Normalization of cost to [0, 1]

A common method used to ensure that a fuzzy or neural system quickly attains more accuracy is to somehow reduce the data set so that only the most important information is given to the network, while all other data are eliminated so as not to confuse the system. Unfortunately, there does not seem to be a large amount of research in the field of nonlinear dimensionality reduction for sparse data sets. Most of the research found on this topic was related to image processing, which does not suffer from the problem of small data sets as is the case of NASA's article test data. We use only PCA and locally linear embedding (LLE), discussed in the next subsections, but other methods are available.

The Isomap (isometric feature mapping) method, developed by Tenenbaum, Silva and Langford (2000), is a nonlinear dimensionality reduction method that has been applied to image processing. This algorithm attempts to use classical multidimensional scaling (MDS) to map data points from a high-dimensional input space into low-dimensional coordinates of a nonlinear manifold (Gering, 2003) by working within neighborhoods. The Isomap method, as well as the LLE, relies heavily on the nearest neighbor algorithm. Most nonlinear dimensionality reduction methods (Brand, 2003; Demartines & Herault, 1997; Friedrich, 2003; Gering, 2002; Roweis & Saul, 2000) require some sort of nearest neighbor processing. Once again, this is not viable for use in extremely small data sets. We simply do not have enough data to make a good neighborhood grouping. However, in order to exemplify the problem, we do present the LLE algorithm and the results obtained using LLE prior to ANFIS processing.

An overall processing block diagram with LLE preprocessing is shown in Figure 1. The LLE processing block should be replaced by a PCA block when principal components decomposition is used. The first normalizing procedure is applied before the transformation is computed. The raw data are normalized to have a mean of zero and a standard deviation of one. The second normalization procedure is applied to the transformed (either by PCA or by LLE) data before they are fed to the fuzzy-neuro system. This normalization step ensures that the input data's range is in the range [-1, 1]. Often we also normalized the cost to the range [0, 1].

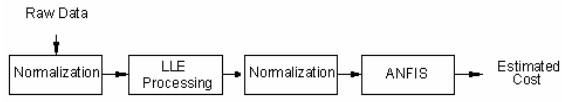
We try both the locally linear embedding (LLE) algorithm (Roweis & Saul, 2000; Saul & Roweis, 2003) and principal components analysis (PCA; Cohen, 1998) to reduce the dimensionality of the data set, which is then used to train an ANFIS to predict the cost of engine tests. PCA is a linear operation, however, and this system is highly nonlinear. LLE is a nonlinear method of reducing the dimensionality of the data set and we therefore expected it to produce better results than PCA; this was not proven to be the case during testing. Nonetheless, we believe that the LLE method would yield good results if a large data set were available, so that better neighborhoods could be defined.

Locally Linear Embedding (LLE)

Locally linear embedding, developed by Roweis and Saul (Roweis & Saul, 2000; Saul & Roweis, 2003), is a nonlinear dimensionality reduction method originally applied to image processing. Liou and Kuo (2002) applied LLE to visualization of economic statistics data. We implemented the LLE method for nonlinear dimensionality reduction of input data for engine test cost estimation. A fuzzy system was then developed which predicts the engine test cost based solely on the reduced data, as shown in Figure 1. LLE attempts to map the input data to a lower dimensional global coordinate system that preserves the relationships between neighboring points (Gering, 2003). Locally, linear neighborhoods of the input data are then mapped into a lower dimensional coordinate system. Unfortunately, it is very difficult to work with neighborhoods when the size of the data set is as small as ours. However, one of the purposes of this chapter is to present ways of performing accurate cost estimates for general applications, and this method might prove useful to readers who have sets of data composed of many exemplars.

The LLE algorithm is divided into three steps: selection of neighbors; computation of weights that best reconstruct each data point by its neighbors; and mapping to embedded coordinates (Friedrich, 2002; Roweis & Saul, 2000). The first step simply involves finding K nearest neighbors. We accomplish this by finding Euclidean distances or finding all neighbors within a fixed radius. The reconstruction weights are determined by minimization of a cost function. The data consist of real-valued vectors, each of dimensionality sampled from an underlying manifold. As long as there are enough sample points, it is expected that each data point lies on or close to a locally linear section

Figure 1. Block diagram of complete ANFIS system, including pre-processing



on the manifold. The local area is then characterized by linear coefficients that reconstruct each data point from its neighbors. The reconstructed errors are measured by

$$\varepsilon(W) = \sum_i \left| X_i - \sum_j W_{ij} X_j \right|^2 \quad (1)$$

This cost function adds up the squared distances between all of the data points, X_i and their reconstructions $W_{ij}X_j$. The weights represent the contribution of the j^{th} data point to the reconstruction of the i^{th} data point. The weights are computed by minimizing the cost function on two conditions: (a) each data point is reconstructed only from its neighbors, and (b) the cost function is minimized so that the rows of W sum to one. For any particular data point, these weights are invariant to rotations, rescalings, and translations of that data point from its neighbors, meaning that these weights reflect intrinsic geometric properties of each neighborhood (Saul & Roweis, 2003).

The final step in the LLE algorithm is mapping the high-dimensional data, X, to the new lower dimensional space coordinates, Y. Each high dimensional data point is mapped to the lower dimensional vector representing the embedding coordinates. The embedding coordinates, Y, are obtained by, once again, minimizing an embedding cost function

$$\Phi(Y) = \sum_i \left| Y_i - \sum_j W_{ij} Y_j \right|^2 \quad (2)$$

As with the previous function, (2) is based on locally linear reconstruction errors, but the weights are now fixed while Φ is optimized. This cost function can be manipulated into a quadratic form and minimized by solving a sparse N×N eigenvalue problem whose largest d nonzero eigenvectors provide the set of orthogonal coordinates centered on the origin, where d is the desired reduced dimension size. Pseudocode for implementing the LLE algorithm is given in Saul and Roweis (2003) and will not be repeated here.

The LLE-reduced data are fed to the LLE-ANFIS system and are not used for the other systems in our suite.

Principal Component Analysis (PCA)

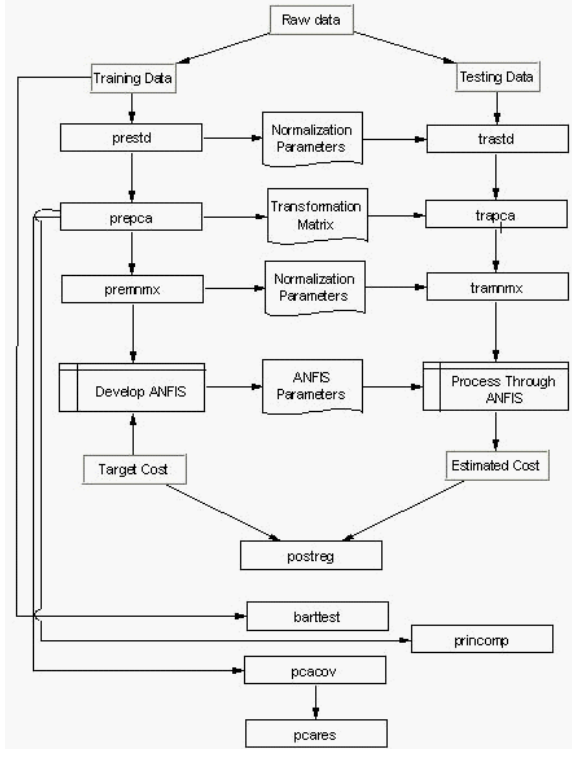
We also applied principal component analysis to reduce the dimensionality of the data set (Kaminsky, 2000). The only disadvantage of using PCA in this situation is that PCA is a linear transformation and the data has a highly nonlinear relationship between individual data components. This is why using a nonlinear dimensionality reduction method, such as LLE, was thought to be a favorable alternative to using PCA. The nonlinearities, however, are addressed by the nonlinear ANFIS and neural systems to which the PCAs are fed. Principal component transformation proved to be a powerful preprocessing technique when applied to the normalized input data. There are two reasons why we are performing a PCA: to reduce data dimensionality (because we have an extremely small number of test articles) and to gain a further understanding of the relative importance and information content of the input data collected. This might yield

insight into the data collection process itself, indicating redundant variables and, possibly, a need for other types of data input.

The main idea behind PCA is to (linearly) transform the original input data set into a different set which contains uncorrelated data. Principal component analysis uses singular value decomposition to compute the principal components of a set of data. The transformed vectors' components are uncorrelated and ordered according to the magnitude of their variance (Kaminsky, Rana & Miller, 1993). The new set, then, is ordered such that the first column contains the most informative data (as measured by variance), and the last column contains the least important data. This allows us to remove the last few columns of data, therefore reducing the dimensionality, while discarding as little information as possible (Cohen, 1988; Kaminsky, 2000). So by choosing only the first few principal components that influence the variance the most, we orthogonalize the input data, while eliminating vector components that contribute little to variations in the data set.

The principal components, or a normalized version of these, are the inputs to the fuzzy system PCA-ANFIS. Figure 2 shows a block diagram of the PCA-ANFIS system, with the main processing routines indicated in the rectangular blocks by the appropriate Matlab commands. Most of the figure shows processing routines; the "postreg" block

Figure 2. Block diagram of the process used in developing the PCA-ANFIS systems. Routines are shown by their Matlab commands.



on down, shows analyses routines that are not used during regular processing. The block labeled ANFIS is discussed in detail elsewhere in this chapter.

Let us denote the original data, in our case the engine test data, by x. First, we compute the mean vector of the measurements, μ_x, and the covariance matrix, S_x. The eigenvalues, λ, of S_x are then computed. An orthonormal matrix U is made from the eigenvectors of S_x so that

$$L = U^T S_x U \tag{3}$$

where L is a diagonal matrix with the vector λ in the diagonal. The original vector, x, is transformed into its principal components, y, by:

$$y = U^T (x - \mu_x) \tag{4}$$

The most important (top rows) of the resulting principal components, y, are the inputs to the ANFIS system. The complete training data set (using all the variables listed in Table 1) was transformed using principal component analysis (PCA). This PCA indicates that the top six principal components (i.e., the six that contribute most to the overall variance in the cost estimate) provide a total of about three quarters of the information for engine tests, as indicated in Table 3. We see that even the most informative component of engine data only really contains between one fifth and one fourth of the total information available in the complete data set. Also, the second component of engine data is almost as "principal" as the first PC, and the third and fourth are, again, similar in information content to each other. Components 5 through 18 are much less important, although that set still contains a cumulative 33% of the total information for engine tests. Components 7 though 18 contain 27% of the information, slightly more than the first component alone, but were always discarded to reduce the dimensionality of the system.

We also obtained the eigenvalues of the covariance matrix of the normalized data, the Z-scores, and Hotelling's T-squared statistic for each data point. Hotelling's T^2 is a measure of the multivariate distance of each observation from the center of the data set. The eigenvalues and T^2 values are listed in Table 4.

The data shown in the column labeled Eigenvalues shows the value of the eigenvalue of the covariance matrix of the data and should be associated with each principal component. This, again, indicates that the first six principal components are important. For example, the largest eigenvalue is 4.5, followed by 2.4, which gives an idea of the relative importance of the principal components. The second data set, shown in

Table 3. Principal component analysis results for engine tests

PC No.	*Information (%)*	*Cumulative Information (%)*
1	22	22
2	21	43
3	13	56
4	11	67
5	4	71
6	2	73

Table 4. Covariance eigenvalues and T-square statistics of engine test data

PC No.	*Eigenvalues*	*Article No.*	*T^2 statistic*
1	4.5079	1	55.4088
2	2.4056	2	114.2291
3	1.7834	3	20.8178
4	1.4666	4	11.0769
5	0.6118	5	13.8477
6	0.5810	6	12.6355
7	0.3711	7	18.3934
8	0.1950	8	24.8131
9	0.1271	9	21.4780
10	0.0291	10	11.0796
11	0.0013	11	137.3189

the column labeled T^2 Statistic, is related to the data set itself (the engine articles), and gives an indication of the position of the data point within the set. The largest *T*-squared value, 137.32, indicates that this data point is very far from the mean or the center of the cluster of test data; this last article, as well as article 2, might be considered "outliers" and clearly have no close neighbors.

ADAPTIVE NETWORK-BASED FUZZY INFERENCE SYSTEMS

Adaptive network-based fuzzy inference systems (ANFIS) were first presented in Jang (1993) and Jang and Sun (1995). These systems combine the advantages of neural networks and fuzzy systems, generating fuzzy inference systems whose membership functions are trained using neural networks to produce the best results. Input–output mapping is therefore based on expert knowledge and training data. Highly nonlinear systems may be created using ANFIS theory.

Standard fuzzy inference systems (FIS) employ "if-then" rules in a noncrisp form (i.e., without using precise quantitative analyses), through the use of membership functions (Zadeh, 1965, 1968, 1978). ANFIS further tune the membership functions to maximize the system's performance. All our ANFIS used Gaussian-type curves for the membership functions; these include the two-sided Gaussian curve membership function (gauss2mf), the Gaussian curve membership function (gaussmf), and the generalized bell curve (gbellmf) membership function.

Our networks are of the type derived by Takagi and Sugeno (1983, 1985), with fuzzy sets only in the premise part (i.e., in the "if" part, not the "then" part). The membership function characterizes the linguistic label in the premise, while a nonfuzzy variable is used in the consequent.

The adaptive network within ANFIS is a multilayer feedforward network that adapts its weights to minimize an error criterion using a gradient search method such as the least mean squares (LMS) algorithm. Adaptation is performed for as many epochs as needed to reach the error criterion. Convergence was always achieved in fewer than 50 epochs.

In a sense, database mining applications such as this one, involve semiautomatic data analysis methods that help users discover some nontrivial knowledge. This knowledge is, in this case, the nonlinear relationship between several input parameters

that describe the engines being tested (raw data from PRDs), and the actual cost[3] of performing the test of the article.

In its Matlab implementation, ANFIS is a training routine for Sugeno-type FIS based on adaptive generalized neural networks. ANFIS uses a hybrid-learning algorithm to identify parameters. It applies a combination of the least-squares (LS) method and the backpropagation gradient descent algorithm for training FIS membership function parameters to emulate a given training data set.

ANFIS SYSTEMS FOR COST PREDICTION OF ENGINE TESTS

Most of the ANFIS systems were developed using grid partition for the generation of the single-output Sugeno-type fuzzy inference system (FIS). We found, when working with the engine tests, that the results from grid partitioning were far superior to those from clustering. This is reasonable because the number of points is so small, that clustering is nearly impossible. When we tried using clustering with the component tests, for which we have somewhat larger (though still very small) training sets, results were more encouraging (Kaminsky & Douglas, 2003).

Membership functions were developed for each input variable. Fuzzification of all crisp quantities was performed. Initial values of the intervals for continuous linguistic variables were determined by the analysis of histograms and clustering methods. Statistical methods were suitable to select relevant features and provide initial intervals defining linguistic variables. Optimization of these initial rules (i.e., optimal intervals and other adaptive parameters) was done by maximizing the predictive (modeling) power of the system using neural networks.

We have also developed "parallel/cascaded" ANFIS: systems consisting of between 2 and 5 ANFIS in the first stage, each of which will concentrate on a subset of inputs and produce their best estimate of cost. A final "merging" of the results of the first stage parallel ANFIS is performed by a second stage (cascaded) ANFIS, or by a feed-forward neural network, which produces the final estimate of the cost. A graphical depiction of the general concept of a Parallel-ANFIS system is shown in Figure 3.

The Parallel-ANFIS system that we selected as prototype consists of two subsystems in the first stage, each with four inputs and one output. The diagram of this Parallel-ANFIS system is shown in Figure 4. We see that the first of the two parallel ANFIS (HEgrid42) uses two membership functions for all inputs, while the second uses 2, 2, 3, and 4 for the number of membership functions. The membership functions are of the gaussmf or gbellmf types. The final ANFIS, which takes the outputs of the first-stage ANFIS as it inputs, uses 3 membership functions of the Gauss2mf type to produce the final estimate of the cost.

We have developed and tested a large number of ANFIS systems. These various fuzzy systems use different number of inputs, different types of inputs, and various number and types of membership functions. The preprocessing applied, and the second stage, if used, also varied. Only a few of the systems developed, those selected to be delivered due to their performance, are discussed here.

Figure 3. Block diagram of the parallel/cascaded ANFIS system

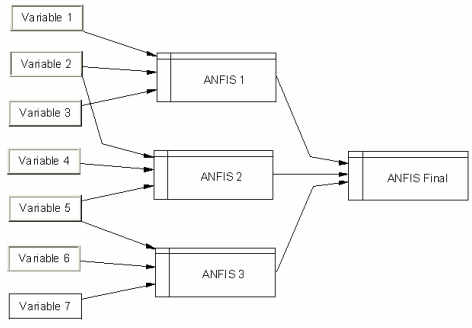


Figure 4. Block diagram of Parallel-ANFIS system showing the two parallel ANFIS systems and the final ANFIS stage

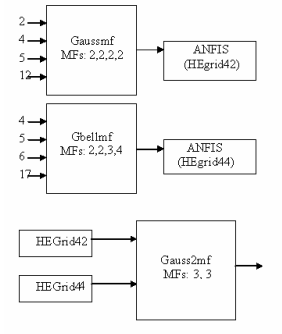


Table 5. Fuzzy/neuro systems for cost prediction of engine tests

ANFIS System	***Input variables***	***No. MFs***	***Comments***
ANFIS	1,3,7,17	3,2,8,2	Hybrid optimization
PCA	1-8,10-12,17,18	Produces PCs	For PCA-ANFIS
PCA-ANFIS	PC1-PC4	4,3,2,4	Gaussmf
Parallel-ANFIS	See Fig. 4	3,3	Gauss2mf & Gbellmf
Parallel-ANFIS-NN1	5,7,13-15,17	2-4	Imputation; 2 input, gaussmf, logsig
Parallel-ANFIS-NN2	1,3-5,7,13-15,17	2-4	Imputation; 3 input, gaussmf/gbell, logsig
LLE-ANFIS	1,3,5,7,13-15,17	4,3,3	Imputation, k=3, d=3
RBFN	All	6	k=6, p=4

In order to present our results in an orderly manner and discuss each of the prototypes, we first tabulate all the systems in Table 5. The simplest system in the suite, a single one-stage ANFIS system, is labeled ANFIS. PCA systems are not actual FIS, but they are systems that produce the transformed inputs to the PCA-ANFIS system. Variables 9, and 13 through16 were not used in computing the PCs. After PCA transformation we discarded all PCs except the first four. The principal components, or a normalized version of these, are the inputs to the fuzzy system PCA-ANFIS.

The Parallel-ANFIS systems use the parallel/cascaded ANFIS implementations as depicted in Figures 3 and 4.

The Parallel-ANFIS-NN systems, depicted in Figure 5, feed the normalized input variables to several ANFIS systems in parallel; these ANFIS produce estimates of the cost which are then fed to a two-layer feedforward backpropagation neural network which produces a final cost estimate by appropriately weighting the various ANFIS cost estimates.

These systems are named parallel-ANFIS-NN1 and parallel-ANFIS-NN2, for double- and triple-input systems, respectively. The Matlab commands traingdx and learngdm were chosen for the training and learning functions of the neural network, respectively. These functions train the network using batch-processing gradient descent with momentum and an adaptive learning rate. This means that for each epoch, if the performance decreases towards the goal, the learning rate is increased; if the performance increases more than a certain factor, the learning rate is decreased and the weight updates are not made. The error criterion used was the sum of squared errors. The number of neurons in the input layer of the neural network was always set equal to the number of inputs to the network which is in turn the number of ANFIS in the previous stage. The output layer consisted of a single neuron. The transfer functions tested were tansig and logsig, smooth, sigmoid-type functions commonly used in neural networks that produce real numbers as output. Figure 6 shows a typical neural network developed for the output stage. The number of membership functions was never allowed to be less than two or more than four. Various initial learning rates were tried with the best results produced with a learning rate μ=0.01, a momentum m=0.3, and logsig as the transfer function for both layers.

LLE-ANFIS feeds data transformed with the LLE algorithm to an ANFIS to predict the cost. We used $d = 3$ as the reduced dimension, $k = 3$ as the number of neighbors, and imputation to fill in gaps in the input data.

The last method used to predict the cost of engine tests is a purely neural solution. It uses radial basis function networks (RBFN; NeuralWare, 1993) to directly predict cost based on the raw data. RBFNs are similar to ANFIS in that they consist of membership

Figure 5. Parallel-ANFIS-NN systems take parallel multiinput ANFIS and feed the predicted cost from each ANFIS to a neural network

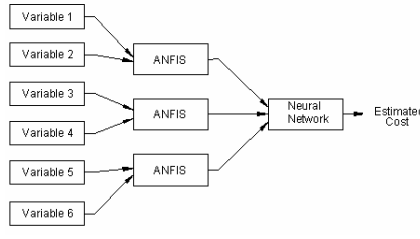
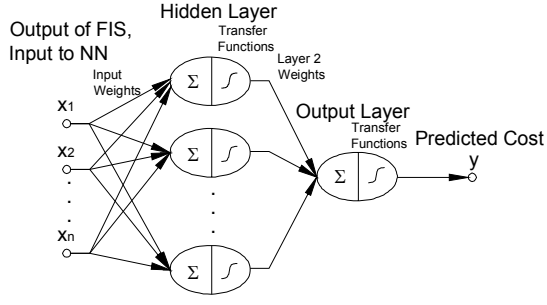


Figure 6. Two-layer feedforward network developed for the Parallel-ANFIS-NN systems

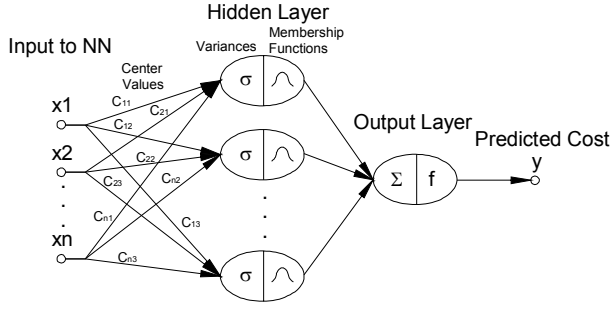


functions that are adjusted through the training stage of a neural network. They typically consist of Gaussian-type transfer functions. First, the centers of the Gaussian functions are found using a *k*-means clustering algorithm on the training data. The *k*-means algorithm groups the data sets into clusters, so that costs are associated with each cluster. After the clusters are found, the *p*-nearest neighbor algorithm is used to determine the width parameter, σ, of the Gaussian transfer function using (5). The respective centers are represented by c_k, where the subscript k represents the cluster of interest and c_{ki} is the center of the i^{th} neighbor. These center values are stored as the neural weights in the input layer.

$$\sigma_k = \sqrt{\frac{1}{p}\sum_{i=1}^{p}\left\|c_k - c_{k_i}\right\|} \tag{5}$$

After the σ parameter of each cluster is determined, the test data can be classified into the appropriate cluster. As with the ANFIS system, a "degree of belonging" to each membership group is obtained. This is done for each article by using (6)

Figure 7. Typical RBNF network used to predict cost of engine tests



$$\phi_k = \exp\left(\frac{-\|x - c_k\|}{\sigma_k^{\,2}}\right) \tag{6}$$

where x is the article data vector whose cost is to be predicted and k denotes the cluster. After (6) is computed, the result is normalized so that the results sum to one. Next, each normalized ϕ_k is multiplied by the calculated average cost of each cluster, and then summed into a single value. This final value is the predicted cost out of the RBFN. Figure 7 illustrates the RBFN system.

The RBFN system was developed using a set of data which applied mean, mode, or median imputation to fill missing variables in some articles. All variables in Table 1 were used.

RESULTS

In what follows we present, separately, the results for each system listed in Table 5. Before presenting the detailed results individually, we discuss the overall, comparative results in a summarized manner. We follow the discussion by particular results for the simple ANFIS, then the PCA-ANFIS, Parallel-ANFIS, Parallel-ANFIS-NN, LLE-ANFIS, and, finally, the RBFN system. In general, as system complexity increases, the accuracy in prediction increases also. We believe that all these systems would prove to be accurate if more data (i.e., more engine tests) were available for training. The performance measures used to evaluate the systems in the cost estimating suite are presented first.

System Evaluation

We would like to evaluate the cost prediction capabilities of the computational intelligence systems developed. There are many ways to measure performance, and it is up to the users to decide, based on their needs and application, which error measurement quantity is most appropriate. Oftentimes the average percentage error or root-mean-squared (RMS) error over all testing articles may be the quantities of interest. In other applications, the maximum error may be more important than an average error. Analysis of the error for each article may indeed be needed in some cases. Is it better to have a 5% error on a very expensive test than a 10% error on an inexpensive test? Clearly the absolute dollar amount should be a consideration. In developing our systems we tried to optimize so that a combination of error measures, those defined in equations (8)-(11) were minimized.

In all following formulas the subscript i denotes the article number and a "hat" over the variable denotes estimate (i.e., the output of the cost estimating system). In order to represent whether the system overestimates or underestimates the cost, the sign is used, with a negative sign indicating that the cost estimate, $\hat{C}_i$, was smaller than the actual cost, C_i. The relative error for each article is denoted by e_i, and the difference between actual and estimated cost is denoted by d_i. The error measures used are listed in what follows:

- Article cost difference

$$d_i = -(C_i - \hat{C}_i) \tag{7}$$

- Article error

$$e_i = \frac{d_i}{C_i} \tag{8}$$

This relative error is usually given as a percentage by multiplying (8) times 100. Clearly, if the estimated cost is smaller than the actual cost, the error in (8) is negative indicating we have underestimated the cost.

- Average error

$$E = \frac{1}{N}\sum_{i=1}^{N} e_i \tag{9}$$

- Average absolute error

$$S = \frac{1}{N}\sum_{i=1}^{N} |e_i| \tag{10}$$

- RMS error

$$E_{RMS} = \frac{1}{N}\sqrt{\sum_{i=1}^{N} e_i^{\,2}} \tag{11}$$

We also use (12), which gives a good indication of the dollar amount by which the total cost differs from the total estimated cost over all tests. This might be a better measure to use in selecting systems for cost estimation than the most frequently used average absolute error and RMS error measures from (10) and (11). The relative error measure weighs the errors more heavily for the expensive items, while the standard error measure weighs all errors by the same amount. We use a subscript of *R* for the relative error measure:

- Relative total error

$$E_R = \frac{\sum_{i=1}^{N} d_i}{\sum_{i=1}^{N} C_i} \tag{12}$$

We also compute the maximum and minimum article errors which can be used to understand the range of errors obtained:

- Maximum absolute percentage error

$$e_{\max} = 100 \max_i \left(|e_i|\right) \tag{13}$$

Table 6. Summary of quantitative results (testing only) for all systems

ANFIS System	*E %*	*S %*	E_{RMS} %	E_R %	e_{min} (%)	e_{max} %
ANFIS	1.0	37.4	18.9	2.1	4.8	64.7
PCA-ANFIS	-0.7	20.6	13.4	-36.8	0.0	63.9
Parallel-ANFIS	-28.6	44.8	23.8	-71.1	10.6	97.7
Parallel-ANFIS-NN1	-16.9	19.9	11.0	-9.11	6.0	30.3
Parallel-ANFIS-NN2	-2.5	7.0	3.9	0.8	1.4	9.7
LLE-ANFIS	-50.9	50.9	28.6	-61.3	11.4	80.8
RBFN	-9.9	14.6	11.7	-22.1	0.8	45.9
CEM[4]	-42.8	42.8	15.1	56.2	7.41	78.93

- Minimum absolute percentage error

$$e_{\max} = 100 \min_i \left(|e_i| \right) \tag{14}$$

Summary of Results

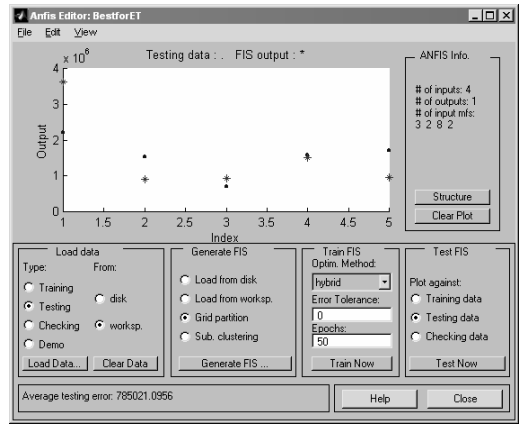
Table 6 presents summary results for all the systems discussed in this chapter; it also includes the evaluation of the cost estimating model (CEM; Lockheed Martin Stennis Operations, 2000; Rocket Propulsion Testing Lead Center, 1997, 1998). We do not know which engines were used for the development of CEM (i.e., we do not know what the training data were). The CEM results were therefore obtained on the entire set of 11 articles which almost certainly includes some, if not all, of the articles used to develop the cost estimating model and relationships used.

All errors are given as percentages. The first numerical column shows the average error from (9). This column could be misleading because overestimates tend to be cancelled by underestimates; in absolute dollar terms, however, this may indeed be desired. The absolute percentage error, computed by (10) may be preferable, and is shown in the column labeled S%. The data shown under the E_{RMS} heading, from (11), are probably the most widely accepted measure of accuracy. Under E_R we list the relative total error from (12). Error ranges are given by the last two columns.

All these measures indicate that the LLE-based system is the poorest performer. The best system is also the most complex one, the parallel-ANFIS-NN2 system with uses ANFIS with three inputs each, followed by a two-layer neural network. For this system the worst case error was less than 10%, while on an RMS sense, the errors were less than 4%. The PCA-ANFIS and Parallel-ANFIS-NN1 systems also produce very good results overall. The maximum error usually happened for one of the "outlier" (very expensive) tests.

Results were obtained both for training and for testing. The training results tell us how well the system has adapted to the "known" input data (i.e., the data that generated the configuration). This clearly should have a low error, or training should continue. Nonetheless, we wish the system to work well with "new and unseen" data (the testing set). If the network is allowed to train far too long and too well for the training set, it will tend to memorize these data, and will not be capable of generalizing to new data (Kaminsky et al., 1993). A compromise between memorization and generalization was sought. In all cases, the training results were excellent, with negligible error (much lower than 1% or on the order of a few dollars). This means that all systems developed learned the input–output relationships for the training data very well. Because all training results

Figure 8. Testing results for the simple ANFIS system using four input variables



were very good and basically identical, regardless of the system used, we will not present detailed training results. These training results were used to determine the number of epochs needed for convergence of the batch processing ANFIS and NN algorithms; all systems converged in fewer than 50 epochs, with many converging in between 10 and 30 epochs.

Simple ANFIS

Figure 8 shows the testing results obtained for the simplest system, named ANFIS. In Figure 8 and on the plots that follow, the actual cost of testing the engines is shown by the dot, while the asterisk indicates the predicted value obtained with the ANFIS prototype. Remember that these 5 articles have not been seen by the network (i.e., these data were not used in the development of the systems). Clearly, only article 1 is poorly estimated (about 65% over the actual amount), while all other unseen articles are estimated with values close to their actual cost, certainly in rough order of magnitude (ROM) which is what we are after. In particular, the third and fourth testing articles have negligible estimate errors (within a few percent). The total average RMS (root mean squared) error is just under $800,000 (see bottom of Figure 8). ANFIS information is presented on the right side of the figure. In this case we see that four input variables were used with 3, 2, 8 and 2 membership functions, respectively. A single output, cost, is produced by the system. If a quick rough order of magnitude (ROM) estimate of the cost is desired, this very simple system might be preferred because it only uses a few data variables that are easily collected and present in all of NASA's test articles and works extremely fast.

PCA-ANFIS System

The results obtained using PCA on the engine test data have also been very encouraging. We obtained an RMS error of less than 500,000, but for the expensive article

we still had an error of about 64%. The errors are very small for all the other articles. The results shown in Table 6 are for the original set of five testing articles. To see how the size of the training set influences results, we included a few more articles in the training set, therefore removing instances from the testing set. These extra training articles were chosen because they are scattered within the possible cost range. Doing this drastically reduced the error of the system. Clearly a significant number of articles must be available in order to be able to compute the principal component transformation matrix. We suggest that this method might be very well suited for cost prediction when a sizable training set is available

Parallel-ANFIS

The testing results obtained for the five engine test articles unseen by the Parallel-ANFIS system were not very accurate. As is almost always the case, the very expensive, high thrust engine is underestimated by a large amount, yielding an average error larger than acceptable. The other articles are all estimated with an error of approximately 10%.

Parallel ANFIS-NN

In the Parallel ANFIS-NN prototypes, several ANFIS systems work in parallel and feed their first-stage cost estimates to a neural network that merges these first-stage estimates and produces as output the final estimate of the cost. We developed and discuss here systems where each ANFIS simultaneously takes either two or three variables as inputs, namely Parallel-ANFIS-NN1 and Parallel-ANFIS-NN2. Once again, the best results were always obtained by using Gaussian type membership functions, either gaussmf, gauss2mf, or gbell in Matlab's language. The neural networks developed for the two- and three-input ANFIS were very similar to each other, and both use logsig for the transfer functions in both layers of the backprop networks.

Parallel ANFIS-NN1

Two inputs are fed to each of the four parallel ANFIS whose outputs were combined by a feed-forward backpropagation trained neural network which produced the final predicted cost. The input pairs to the first stage ANFIS are FuelFlow and Thrust, TestStand and thrust, TestDurMax and PressurantPr, and FuelFlow and OxidizerFl for ANFIS 1 through 4, respectively (refer to Table 1). We used imputation (mean, median, or mode) to fill in values for missing quantities. The variables paired in the double ANFIS used in Parallel-ANFIS-NN1 were chosen by examining the results of the single input ANFIS and choosing variables that complemented each other. For example, if one variable tends to over estimate the cost then another variable that tends to underestimate the cost would be paired with it. Several combinations of variables were tried and these four selected ANFIS produced the best results. The variable Thrust was paired twice in the double input ANFIS because it was one of the most predictive variables.

Once again, we varied the number of membership functions and their types. Gaussian membership functions worked best and the number of membership functions was always maintained between two and four. All costs were normalized to the range [0, 1] before training and testing. The intermediate prediction for each of the 4 parallel 2-

Table 7. Triple input ANFIS (first stage) testing results for each ANFIS, prior to neural network

1st Stage ANFIS Inputs	*Average %*	*RMS %*	*Min %*	*Max %*
TestStand, Thrust, DuratDd	-2.91	2.33	1.15	9.01
TestDurMax, PressuraPr, Pressurant	-9.74	15.98	26.48	36.85
FuelFlow, OidizerFl, Fuel	8.73	10.85	6.39	31.42

input ANFIS were combined by the neural network to produce the testing results shown in Table 6. That is, an average of about 17% underestimate of cost, 11% RMS error, a minimum error of about 6 percentage points, and a maximum error as large as -30%. We think it is important to note that the average sum of differences between predictions and actual costs is only $371,000 for a total cost of $10 million.

Parallel ANFIS-NN2

The ANFIS developed with three inputs achieved excellent results. Different set of triplets of inputs were tested. We selected the first stage ANFIS as shown in Table 7 where the results of each of the three first stage ANFIS (i.e., prior to the neural network merging, are also shown). The ANFIS that used TestStand, Thrust, and DuratDd attained such good results that it could stand alone as a predictor without the neural network stage

Figure 9. Normalized cost results of first-stage ANFIS using the three inputs TestStand, Thrust, and DuratDd along with a gbell membership function

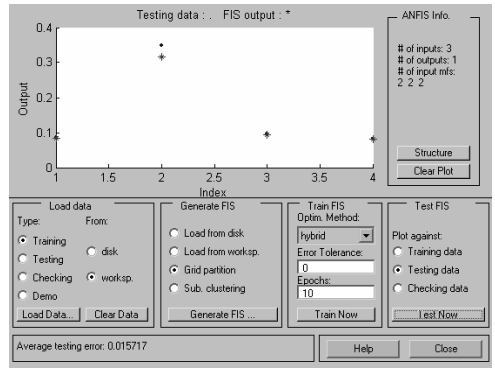


Table 8. Results of the Parallel-ANFIS-NN2 consisting of three triple-input ANFIS cascaded with a two-layer feedforward neural network

NN Cost ($)	*Actual Cost ($)*	*Overall Averages*
1 441 500	1 590 000	% error -2.55
5 306 200	4 935 000	RMS 3.88
1 546 200	1 713 000	S 6.99
1 562 100	1 541 000	

(see Figure 9). The FuelFlow, OxidizerFl, and Fuel ANFIS also attained very good results, even though the variables individually were not the most predictive. Our training results for the first stage (before the neural net) included a few significant errors.

Table 8 presents the neural network output results of Parallel-ANFIS-NN2 for each of the four engines in the testing set, as well as overall evaluation information. This network takes the first stage ANFIS (shown on Table 7) and merges the three estimates into a single final cost estimate. All training errors were well below 1% and all testing errors were below 10%.

LLE-ANFIS System

In this method, the LLE algorithm was used to reduce the dimensionality of the normalized data set. The new, transformed data set was then put into an ANFIS to finally predict the cost of the engine tests. The weakness of this method lies in the LLE algorithm's reliance on the k-nearest neighbor algorithm during the first step which was difficult to accomplish due to the extremely small number of points in the data set we utilized. An ANFIS was then developed from the new lower dimensional data, using grid partitioning and a linear output membership function. Several trials were performed to develop the best ANFIS by varying the number and type of membership functions. The best results were obtained using the set of eight variables shown in Table 5. We experimented with designs using different number of clusters, k, and also various LLE-reduced dimensions, d. Finally, we used $k=d=3$. The best results were obtained with a gauss2mf membership function of size 4, 3, and 3, for each transformed variable, respectively. The LLE-ANFIS system learned the training set very well, with no error, but was unable to produce good results for unseen articles. The results attained still have an average percentage error of around 66%. The first two testing set articles are both estimated to cost much less than what they actually cost to test. Interestingly, all articles' costs were underestimated, producing an estimate considerably lower than the actual cost of performing the test. This also happened when using CEM.

Radial Basis Function Network

The final method discussed uses a radial basis function network (RBFN) to predict the engine test cost directly from the raw data. Results were encouraging when we used all data available for training, but we must remember that the k-means algorithm is used for training and, as we have stated often, the training set is much too small to expect any neighborhood-based algorithm to perform well. Nonetheless, we present the results because they are certainly worth pursuing for cases where larger sets of training data are available, as is also the case for the LLE-ANFIS system.

The RBFN was developed by varying the number of clusters, k, and the number of nearest neighbors, p. The number of inputs was also varied, but the results were best

Table 9. Results for the RBFN cost estimator

RBFN Cost	***Actual Cost***	***Overall Averages***	
1 542 400	1 590 000	% error	-9.87
2 668 900	4 935 000	RMS	11.71
1 726 800	1 713 000	S	14.59
1 674 000	1 541 000		

when the full data set of eighteen variables was used with imputation algorithms used to fill in missing values. The best results were attained when both k and p were set to the maximum values of 6 and 4, respectively. Table 9 presents the predicted and actual costs for the test articles as obtained by the RBFN. The predicted cost is a weighted average of the prototype cluster costs, with the weights given by a measure of the distance to each cluster, as given by (6).

The RBFN predicted the cost of articles 1, 3 and 4 fairly accurately, but had trouble predicting article 2, the most expensive testing article in this testing set.

Comparison of Results for All Methods

A comparison of the overall results was given at the beginning of this section, in Table 6. A graphical representation of the results is given in Figure 10. The best results were obtained by the Parallel-ANFIS-NN2 system which uses a feedforward backpropagation neural network that takes the costs predicted by each of three three-input ANFIS and combines them into a single output cost. This system achieved a testing average percentage error of -2.5% with no quantity individually estimated with an error above 10%. A few of the methods developed were not effective, namely Parallel-ANFIS and LLE-ANFIS; the latter did a poor job of predicting almost all the articles. Keep in mind that the neural-based systems (the four right-most systems in Figure 10) were trained with 7 articles in the training set while the first three shown were trained with six articles only.

We would have liked to compare all our systems to results obtained with other previously developed systems described in Sundar (2001). However, very few (if any) results are available from those other methods, so strict comparisons cannot be made and final conclusions cannot be drawn. Also, as stressed earlier, many of these complex systems require a level of input data detail which is simply not available prior to

Figure 10. Standard error measures of cost prediction (percentages) for all testing results using computational intelligence methods

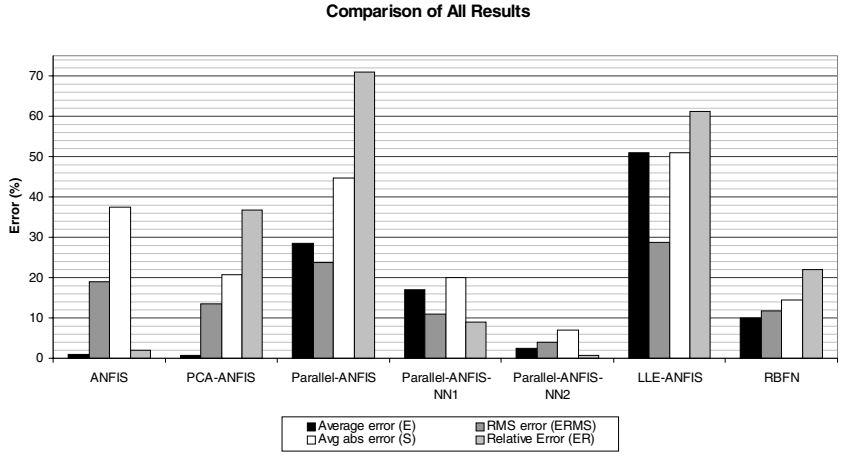
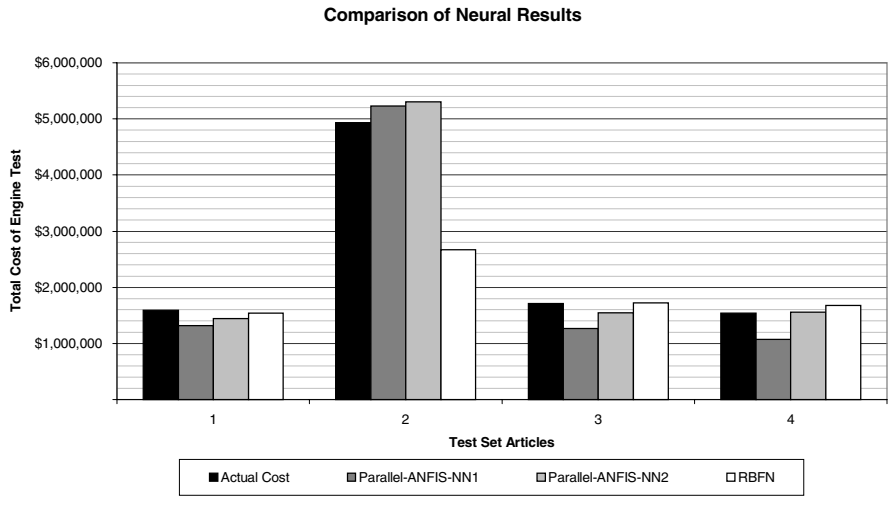


Figure 11. Comparison of all four neural methods showing actual and predicted costs of the articles in the testing set



performing the tests. The CEM system (Lockheed Martin Space Operations, 2001; Lockheed Martin Space Operations, 2000; Rocket Propulsion Testing Lead Center, 1997, 1998) worked relatively well on an RMS sense, but underestimated the cost of most engine tests. The advantage of CEM is that it provides cost estimation relationships that may be quite useful and may be adjusted easily to incorporate temporal data. The other software systems could not be run with the data collected from PRDs.

The neural methods seem to be most promising. In a way, they are similar to the original methods developed at NASA and Lockheed because they seek to establish relationships among variables or functions of these variables. Unfortunately, the relationships obtained with the neural network are "hidden" and coded within the networks' weight matrices. Figure 11 shows the neural results for the four articles in the testing set and the actual costs of performing the tests.

CONCLUSION AND SUGGESTIONS FOR FURTHER WORK

We have developed several computational intelligence systems to predict the cost of testing engine and component tests based on standard data collected by NASA in their project requirement documents (PRD); only engine tests were discussed in this chapter. Our computational intelligence systems take various variables from the PRDs and use adaptive network fuzzy inference systems (ANFIS) and neural networks to combine, in a nonlinear manner, the values of these variables to produce an estimate of the cost to perform engine tests at the Stennis Space Center. Raw data were normalized and, for some

of the systems, transformed with principal component analysis (PCA) or locally linear embedding (LLE) to reduce the systems' dimensionality prior to further processing.

We have also designed "specialized" fuzzy systems that work in parallel, each of which provides an estimate to a final fuzzy or neural stage which combines these results to obtain a final estimate. Our results indicate that an error of around 10%, on the average, may be expected with these parallel ANFIS systems. However, most of the test articles are estimated with considerably less than 10% error.

We have achieved very good results with a very small set of training data. The results of the RBFN, PCA-ANFIS, and both Parallel-ANFIS-NN systems are very accurate. Remember that the desire is to obtain a rough-order-of-magnitude (ROM) estimate for the cost of performing engine tests. The generalization ability of our ANFIS systems has been proven. We conclude that the project was successful at using new artificial intelligence technologies to aid in the planning stages of testing operations at NASA's Stennis Space Center.

A linear transformation—namely PCA—as well as the nonlinear locally linear embedding (LLE) algorithm were used for dimensionality reduction. It could be wise to try other nonlinear transformations on the original data before feeding them to the ANFIS systems.

Coupling the application of fuzzy logic and neural networks for modeling and optimization with the Risk Constraint Optimized Strategic Planning (RCOSP) model of Sundar (2001) is expected to yield more accurate and robust estimation of cost and an understanding of the requirements to provide rocket propulsion testing for the future. CEM, the Cost Estimating model of Lockheed Martin Space Operations (2000, 2001) and Rocket Propulsion Testing Lead Center (1997, 1998), combines complexity factors and cost estimating relationships to predict the approximate cost of performing technology development test programs. At this point, all these software pieces work independently. NASA (2001) presents analysis of PRDs and a tool (DOOR) which uses, updates, and databases PRD data. It would be very beneficial to somehow join DOORS with our cost prediction suite so that PRD data may be passed directly to the prediction systems.

In order to keep the model (decision algorithm) from becoming obsolete, some kind of date information (incremental information) must be associated with it. At the same time, we would like the decision algorithms for similar database mining queries to be reusable. An effort to homogenize all data would be valuable.

Finally, it would be of great use to be able to predict the cost of each of the three main functions that affect cost: modification, design, and fabrication, as CEM does. This can be achieved using the same type of ANFIS and neural networks that we have discussed. Unfortunately no training data are available at this moment to train such systems (i.e., we do not have access to these detailed costs).

ACKNOWLEDGMENT

We wish to thank Lockheed Martin Stennis Operations personnel for their contributions to this project. In particular, we acknowledge their effort in providing us with the required data, documents, and software.

REFERENCES

Brand, M. (2003). Continuous nonlinear dimensionality reduction by kernel eigenmaps. Retrieved November, 2003, from http://www.merl.com/reports/docs/TR2003-21.pdf

Chongfu, H. (1998, May 4-9). Deriving samples from incomplete data. In *Proceedings of the IEEE World Congress on Computational Intelligence,* Anchorage, AK.

Cohen, A. (1988). *Biomedical signal processing. Volume II: Compression and automatic recognition.* Boca Raton, FL: CRC Press.

Danker-McDermot, H. (2004). *A Fuzzy/neural approach to cost prediction with small data sets.* Master's thesis, University of New Orleans, LA.

Demartines, P., & Herault, J. (1997). Curvilinear component analysis: A self-organizing neural network for nonlinear mapping of data sets. *IEEE Transactions on Neural Networks, 8,* 148-154.

Friedrich, T. (2002). *Nonlinear dimensionality reduction with locally linear embedding and isomap.* MSc dissertation, University of Sheffield, UK.

Gering, D. (2003). Linear and nonlinear data dimensionality reduction. Retrieved November, 2003, from http://www.ai.mit.edu/people/gering/areaexam/areaexam.pdf

Granger, E., Rubin, M., Grossberg, S., & Lavoie, P. (2000, July 24-27). Classification of incomplete data using the fuzzy ARTMAP neural network. *Proceedings of the IEEE International Joint Conference on Neural Networks,* Como, Italy.

Ishibuchi, H., Miyazaki, A., & Tanaka, H. (1994, June 27-July 2). Neural-network-based diagnosis systems for incomplete data with missing inputs. *Proceedings of the IEEE International Conference on Neural Networks,* Orlando, FL.

Jang, J. S. (1993). ANFIS: Adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics. 23*(3), 665-684.

Jang, J. S., & Sun, C-T. (1995). Neuro-fuzzy modeling and control. *Proceedings of the IEEE, 83,* 378-405.

Kaminsky, E. J. (2000, June 26-29). Diagnosis of coronary artery disease using principal components and discriminant functions on stress exercise test data. *Proceedings of the 2000 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS 2000),* Las Vegas, NV.

Kaminsky, E. (2002). *Highly accurate cost estimating model (HACEM).* (Final Report, LA BoR No. NASA Stennis Space Center.

Kaminsky, E. J., & Douglas, F. (2003, September 26-30). A fuzzy-neural highly accurate cost estimating model (HAC*E*M). *Proceedings of the 3rd International Workshop on Computational Intelligence in Economics and Finance (CIEF'2003),* Cary, NC.

Kaminsky, E. J., Rana, S., & Miller, D. (1993, September). *Neural network classification of MSS remotely sensed data.* Report CAAC-3930, NASA, Stennis Space Center, MS.

Liou, C.-Y., & Kuo, Y.-T. (2002, November 18-22). Economic states on neuronic maps. *Proceedings of ICONIP 2002, 2* (pp. 787-791). Singapore.

Lockheed Martin Space Operations. (2001, September). NASA/Stennis space center propulsion testing simulation-based cost model. *Ninth International Conference on Neural Information Processing,* Stennis Space Center, MS.

Lockheed Martin Stennis Operations. (2000, September). A cost estimating model (CEM) and cost estimating relationships (CER) validation and evaluation analysis (Version 1). NASA Report, Stennis Space Center, MS.

NASA. (2001, July). *John C. Stennis Space Center preparation of SSC propulsion test directorate (PTD) project requirements document* (Tech. Rep. No. SOI-8080-0004), NASA-SSC. Stennis Space Center, MS.

NeuralWare. (1993). *Neural computing* (Vol. NC). Pittsburgh, PA: NeuralWare.

Rocket Propulsion Testing Lead Center. (1997, June). *A cost estimating model (CEM, Revision 1).* NASA Stennis Space Center, MS.

Rocket Propulsion Testing Lead Center. (1998, March). *A cost estimating model* (Systems Requirement Document SSC-LC-008, Decision Support System).

Roweis, S., & Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science, 290,* 2323-2326.

Saul, L. K., & Roweis, S. T. (2003). *An introduction to locally linear embedding.* Retrieved December, 2003, from http://www.cs.toronto.edu/~roweis/lle/papers-/lleintro.pdf

Sundar, P. (2001). *Bayesian analysis of the RCOSP model.* SFFP Final Report. NASA Stennis Space Center, MS.

Takagi, T., & Sugeno, M. (1983, July 19-21). Derivation of fuzzy control rules from human operator's control actions. *Proceedings Symposium on Fuzzy Information, Knowledge Representation and Decision Analysis (IFAC),* Marseille, France (pp. 55-60).

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man, and Cybernetics, 15,* 116-132.

Tenenbaum, J., Silva, V., & Langford, J. (2000). *A global geometric framework for nonlinear dimensionality reduction science, 290,* 2319-2322.

University of New Orleans. (2000, October). *Project requirements risk analysis* (Department of Mathematics Tech. Rep.). New Orleans, LA.

Zadeh, L. (1965). Fuzzy sets. *Information Control, 8,* 338-353.

Zadeh, L. (1968). Probability measures of fuzzy events. *Journal Math Analysis and Applications. 23,* 421-427.

Zadeh, L. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems. 1,* 3-28.

Zhou, Y. (2000). *Neural network learning from incomplete data.* Retrieved November 2003, from http://www.cs.wustl.edu/~zy/learn.pdf

ENDNOTES

1 Matlab is a trademark of The Mathworks.

2 This work was performed under grant no. NASA(2001)-Stennis-15, "Highly Accurate Cost Estimating Model (HACEM)". The contract is between the University of New Orleans (UNO), Department of Electrical Engineering, and the National Aeronautics and Space Administration (NASA), through the Louisiana Board of Regents (LA-BOR). Access to details and code are available through NASA's Technology Transfer Office: Request Highly Accurate Cost Estimating Model, NASA NTR SSC-00194, May 2003.

3 The dollar amounts used for "actual cost" are not in themselves accurate; they are NASA's posttest estimates.

4 CEM was not developed by the current authors and it is used only for comparison purposes. Errors shown for CEM are for the entire set of 11 articles which may include CEM training data.