# Neural Network Cost Estimating Relationships

Presented
At the
2010 ISPA/SCEA Conference and Training Workshop

By
Edwin B. Dean
Consultant
designforvalue@att.net

## Introduction

This paper is provided to answer questions that were voiced during and after the presentation of Dean (2009).

This paper began in 1961. At that time I was a cooperative student at the Johns Hopkins University Applied Physics Laboratory in Laurel, Maryland. I was designing and testing the logic used to control the Applied Physics Laboratory Automaton – a pre robot. "The Beast" (APLJHU, 1960) was a pre – robot because robots are defined to be controlled by a digital computer. Control of "The Beast" was totally wired logic. Next door, Fred Hiltz was using an analog computer to simulate the electronic action of synapses in nerve cells. I was fascinated by his work and the thought that we might be able to understand how the brain worked. This led to a senior project at Virginia Tech where I designed simulated nerve cells with transistors and used ultrasonics to study feedback and related pulse patterns. My master's thesis in 1965 was a review of various brain models. At the Naval Ordnance Laboratory I used neural networks to analyze signal patterns. In 1987 at the NASA Langley Research Center (LaRC), I discovered the back propagation neural network algorithm (Lippmann, 1987). I coded very crude software to apply it to cost analysis. In 1992, I used neural network software developed by the NASA Johnson Space Center (JSC) to create a neural network launch vehicle cost model (Dean, 1993). During the 2000s I applied neural networks to parametric cost analysis on several projects for Galorath, Inc.

Neural networks provide a valid alternate means of developing cost estimating relationships (CERs). Software exists that allows you to create a neural network CER (NNCER) without any knowledge of mathematics. However, an understanding of the mathematics certainly helps (Lippmann, 1987 and Smith, 1993). This paper does not address the mathematics but is designed to be a road map for using neural networks for estimating cost. The paper addresses pointers to prior use of neural networks for cost analysis, the architecture of a feed forward neural network, how to develop NNCERs, how to compare the goodness of fit of NNCERs with all types of models, how to develop adaptive NNCERs, and how to perform cost risk with NNCERs.

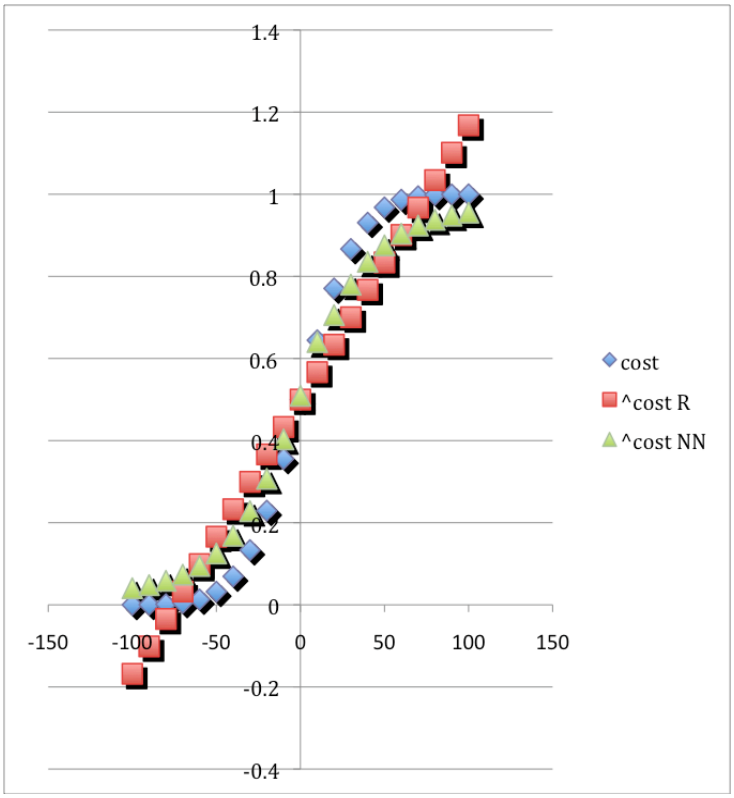
## Prior Application of Neural Networks for Cost Analysis

The primary publications of neural networks applications for cost analysis have occurred within the Association for the Advancement of Cost Engineering International (AACEI) community. When one searches the web based AACEI library using the keyword "neural network", twenty abstracts appear. AACEI members may download most of them for free. The remainder may be purchased.

Dean (2009) summarizes early applications of neural networks for parametric cost analysis, provides two examples of neural network cost estimating relationships, and points to a number of resources for learning about neural networks

## Neural Networks

A neural network is a mathematical entity that simulates the learning capability of the brain. It learns by approximating a set of outputs given a set of inputs. There are many types of neural networks. This paper only addresses the back propagation network discovered independently by Werbos (1974) and Rumelhart, Hinton, and Williams (1986).

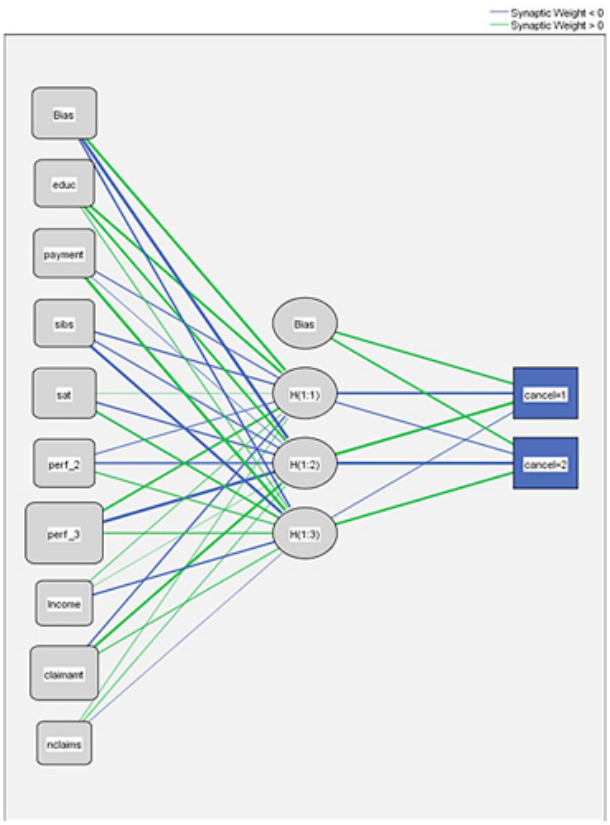



The result of the learning process is a model that predicts the desired outputs based upon a set of (input, output) data. It may have multiple inputs and outputs.

The figure at right demonstrates that the back propagation neural network provides a nonlinear fit to nonlinear data. The blue "cost" diamonds at right are actually points from a cumulative normal distribution S-curve. The brown "^cost R" squares are points derived by applying regression to the "cost" data. The "^cost NN" triangles are the points derived by training the back propagation neural network to the "cost" data. If the neural network had been trained for an additional time, the "^cost NN" triangles would have converged to overlap the "cost" diamonds.

## Neural Network Architecture

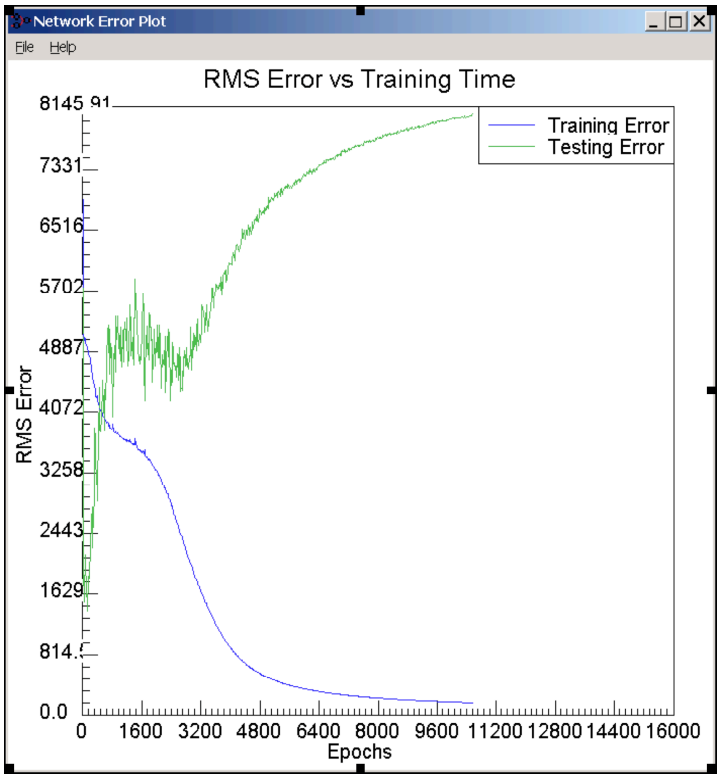
A neural network has an input layer, one or more hidden layers, and an output layer. Each layer contains one or more artificial neurons. For each hidden and output layer, the learning process adjusts the input weights and the threshold weight to reduce the root mean squared output error. The nonlinear activation function enhances the approximation capability.




## Training the Neural Network

A single application of all of the data is called an epoch. The training data set is used to train the neural network. The test data set is used to see how well the neural network generalizes to another independent set of data. The test data set is also used to determine the stopping point for the training. Epochs are applied repeatedly until the inputs provide a reasonable approximation of the outputs.

The figure to the right is a plot of the training data RMS error and the test data RMS error versus the number of epochs applied.



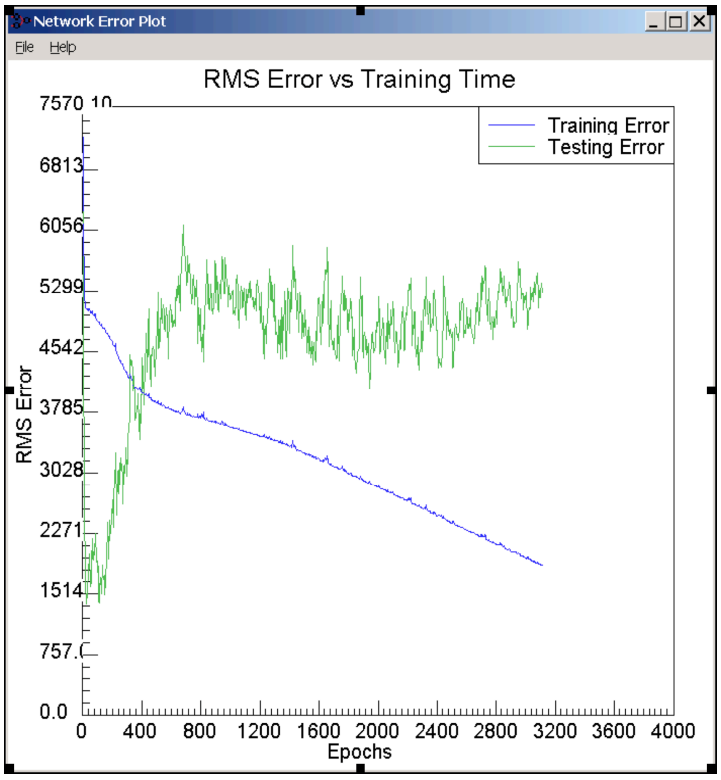

Note, for these data, that the testing error increases as the training data decreases. Note also that the testing error levels off around 4887 for a number of epochs. For the ability to generalize to data not in the training set, it is desirable to reduce the training error as much as possible without increasing the test data error substantially. For this model to maintain a substantial ability to generalize to data of the type in the test data set, the training should be stopped at approximately 3000 epochs. If that generalization capability is not desired, the error of the model with respect to the training data may be reduced substantially.

Note that there is an art to obtaining a good neural network model versus the unthinking permitted when regression is used to generate the model.

**Goodness of Fit**

$R^2$ is used as the goodness of fit measure for regression models. Root mean square error is the goodness of fit measure for the back propagation neural network model.

A goodness of fit measure for different types of predictive models (Dean 2008) is the angle between the predicted data vector and the actual data vector. This is a purely geometric measure that is distribution-independent and method-independent. The smaller the angle is, the better the fit. The cosine of this angle is the equivalent of R for non mean-adjusted data.



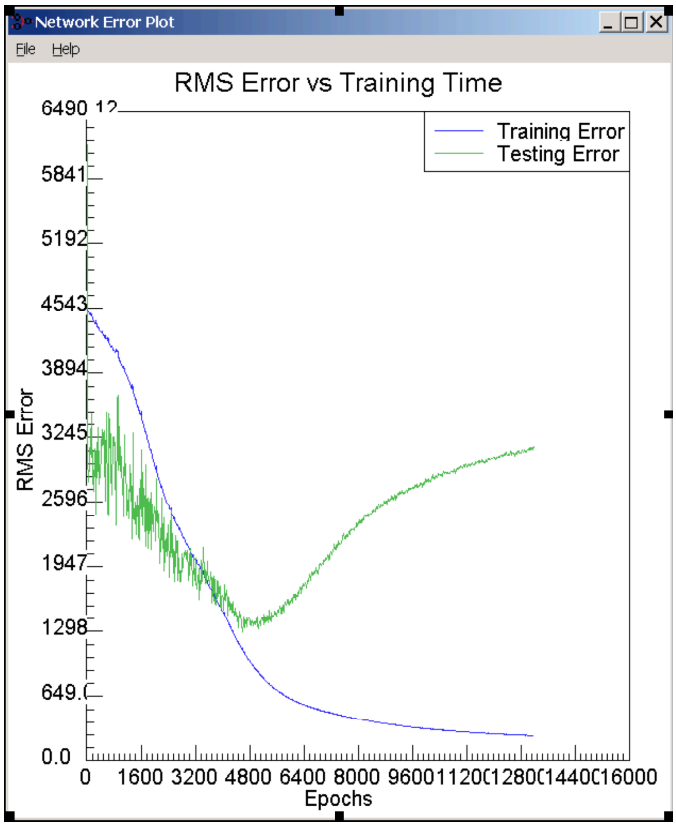

**Generalization Example**

The figure at right is the training/test error plot for a model developed by putting surface-to-air missile data in the training set and ship-to-air missile data in the test set. The training (surface-to-air missile) angle is 13.84 degrees and the test (ship-to-air missile) angle is 8.26 degrees.

The flat test error and declining training indicates that the surface-to-air missile error may be reduced further while mainlining the ability of the model to generalize to ship-to-air missiles.

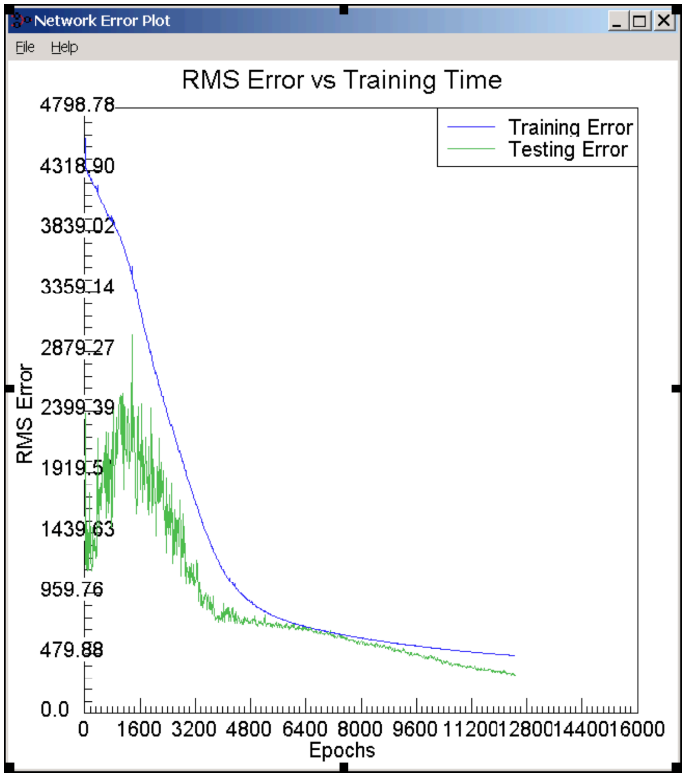
**Adaptive CERs**

An adaptive CER is one that ensures that a set of data points carries more weight in the model than other data points (Book, Broder, and Feldman, 2009). The neural network training process provides a simple way to develop an adaptive CER. Instead of using a random set of data points in the test set, place the points for which you desire the most weight in the test data set. Use the test data set to stop the training as the test data error curve turns up. At that point the neural network has the least error for the data points desired to have the most weight in the manifold fitting process.




The example below and to the right contains all surface-to-air and ship-to-air missiles in training set. All ship-to-air missiles are in test set. The training (all missile) angle is 3.44 degrees and the test (ship-to-air) angle is 3.71 degrees. Four degrees is a very small angle, so this model has excellent predictive qualities.

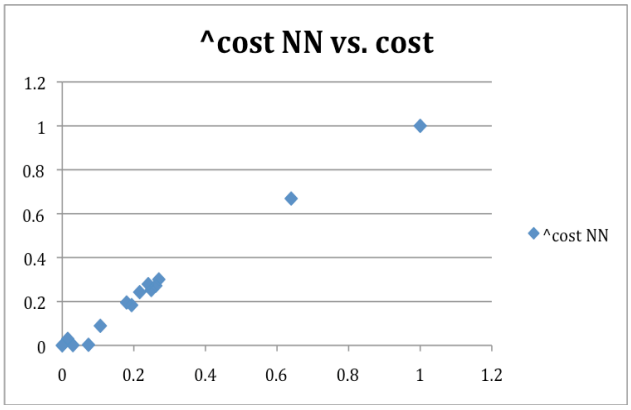
Note that both the training and the test errors are continuing to decrease at this level of training. Note also that the error for both missile types is substantially lower than the above example and that the test error is substantially the same as the training error. Thus the model does a good job of predicting both missile type costs.

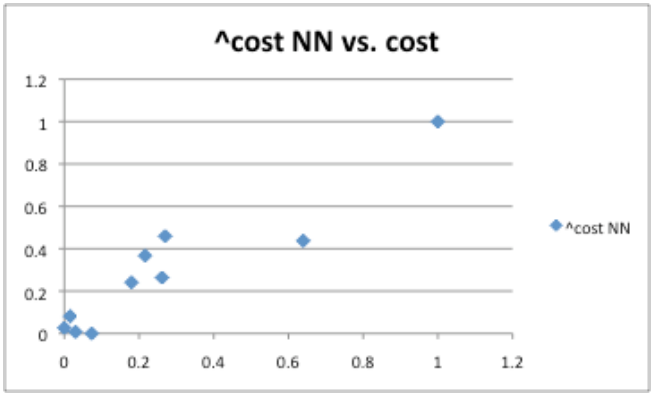



The downside is that it may not generalize to other types of missiles. For that to happen, data for those types of missiles should be included in the test data set.

Note that the data for these examples came from the 1994 version of the Advanced Mission Cost Model (Econ and Cyr, 1994). The parameters (the inputs) are empty weight, payload weight, initial operational capability date, research and development quantity, and production quantity. They were used to predict total project cost in 1989 dollars. There were 10 surface-to-air and 4 ship-to-air missiles. The presented values have been normalized to protect the data.

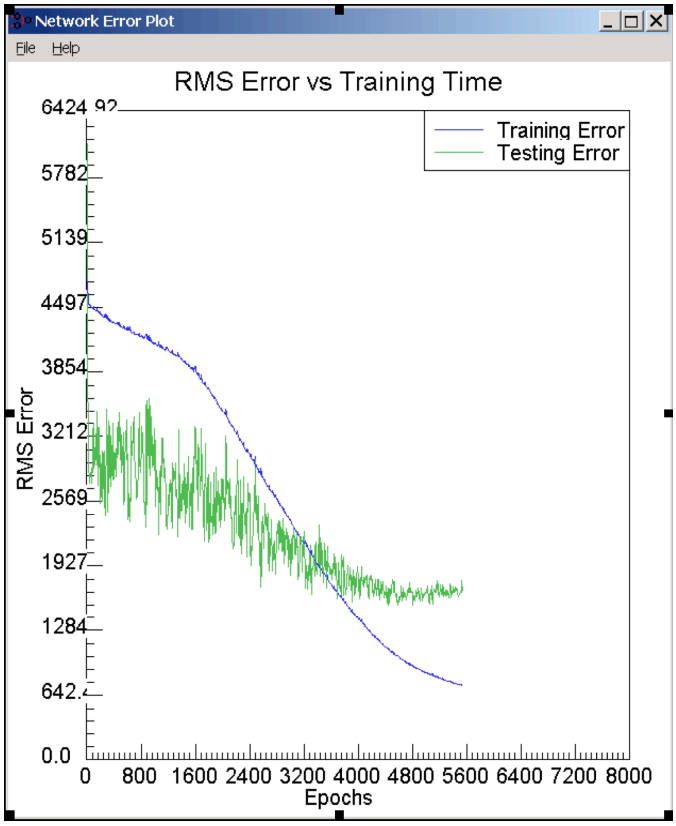
The graph below left provides a visualization of how good a fit of 3.44 degrees is. The graph below right provides a comparison with the generalization example above with a 13.84 degree fit.




**Adaptive CER with Generalization Example**

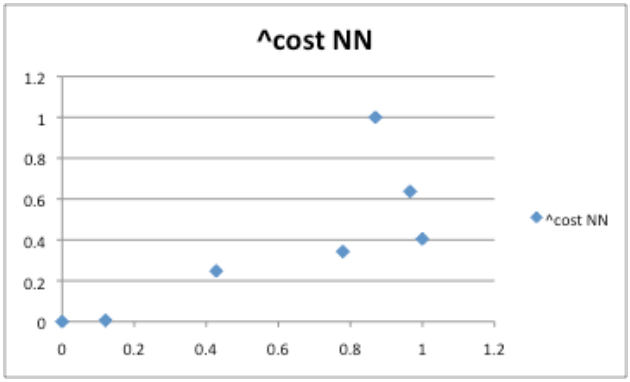
For this example, some surface-to-air missiles and some ship-to-air missiles are in the training set. The remainder of the surface-to-air missiles and all of the ship-to-air missiles are in the test set. The training angle is 5.00 degrees and the test angle is 16.2 degrees. The ship-to-air angle is 8.14 degrees.




The test error is leveling off here and is about to turn up somewhat while the training error is continuing to decline. Thus, continuing to train will damage the ability of the model to generalize. The ship-to-air error is considerably better than the overall generalization error.

The results here are a little fuzzy because there was no clean objective for what data was to be trained on and what data was to be tested. There is an art to determining the composition of the training and test data sets. Note, however, that because the ship-to-air data was in both the training data and the test data sets, it's fit is superior to the total test set.

The graph at right provides a glimpse at how well the fit of 16.2 degrees is. Note that there is big difference between graphs of the fits of 3.44, 13.84 and 16.2 degrees. Pardon my fuzzy numbers, but based upon my experience, fits of 5 degrees or less are excellent, fits between 5 and 10 degrees are pretty good, fits between 10 and 15 degrees are on the edge of acceptability, and fits above 15 degrees are pretty bad.



**Cost Risk Using Neural Networks**

It is unfortunate, but my understanding is that statistics analogous to regression do not yet exist for neural networks. This eliminates using the uncertainty associated with the development of the CER itself. Establishing statistics analogous to regression is a research topic that could provide one or more excellent papers. A hint is the fact that the cosine of the angle between the data vector and the predicted data vector holds the same relation to the non mean-adjusted data as R does to mean-adjusted data.

The void of regression type statistics mandates that one must use what I call push-cost risk. In push-cost risk, probability distributions are placed on the input parameters and are fed as inputs to the trained neural network. This creates a cost risk distribution at the output of the model. The nonlinearity of the model then creates a distribution of a type that the cost community is not yet familiar with – probably from nonlinear statistics (Murray and Rice, 1993). Determining the output distribution type is another area for research. However, the fact remains that the output distribution, be it only numeric, does represent the cost risk.

When subsystems are involved (work breakdown structures), one must use push-cost risk with a network of networks. Each subsystem has a separately trained neural network that is input to the appropriate subsystem output or summing junction or to the system output summing junction.

When activities or processes are involved (process breakdown structures), one must use push-cost risk with a network of networks. Each activity or process has a separately trained neural network that is placed at the appropriate location in a PERT/CPM (Levin and Kirkpatrick, 1966) type network.

Risk register distributions may be used as training inputs or may be attached to any summing junction within a risk simulation.

**Implementing Neural Network Cost Risk**

Good neural network software permits access to the weights of the trained neural network. Some neural network software provides C or C++ code as well as the weights. QuikNet v2.23 by Jensen (2003) is an easy to use and inexpensive neural network software shareware download for Windows. It works well with Windows 2000 and Windows XP. I have not tried it with Windows Vista or Windows 7. However, I suspect it will run well in Windows 7 in XP compatibility mode.

The cost risk network of networks can be implemented in a spreadsheet or as code. Regression based or other types of CERs can be integrated into the cost risk network of networks. For other types of CERs read Meisl (1989).

**Combining Duration and Cost**

Note that neural networks may have multiple outputs. There is no reason that duration cannot be a second output. If so, then cost and schedule may be trained and used simultaneously for estimating and simulating joint cost and duration risk.

**Summary**

This paper demonstrates that neural network CERs are a practical alternative to regression-based means of developing CERs. They provide a non assumption based CER derived from data. The training process is easy to use, although there is an art to the training process. Neural networks also provide a means of developing adaptive CERs, for simulating cost risk, and for simulating cost and duration risk.

This paper provides a roadmap for the application of neural network CERs within parametric cost analysis and estimating.

Resources

AACEI. Association for the Advancement of Cost Engineering International, http://www.aacei.org/.
APLJHU (1961). Johns Hopkins Beast, http://en.wikipedia.org/wiki/Johns_Hopkins_Beast .
Book. S., M. Broder, and D. Feldman (2009). "Statistical Foundations of Adaptive Cost-Estimating Relationships," *Proceedings of the 2009 ISPA/SCEA Professional Development and Training Workshop*, 2-5 June, Saint Louis, MO.
Dean, E. B. (1993). "A Neural Network Expendable Launch Vehicle Cost Model," presented at the *1993 NASA Cost Estimating Symposium*, NASA Glenn Research Center, Cleveland, OH, USA, 13-15 October.
Dean, E. B. (2008). "The Angle: A Goodness of Fit Metric for Cost Analysis", *Proceedings of the ISPA/SCEA 2008 Joint International Conference*, 12-16 May, Noordwijk, The Netherlands.
Dean, E. B. (2009). "Parametric Cost Analysis Using Neural Networks", *Proceedings of the 2009 ISPA/SCEA Professional Development and Training Workshop*, 2-5 June, Saint Louis, MO.
Econ and K. Cyr (1994). Advanced Mission Cost Model, http://cost.jsc.nasa.gov/AMCM.html .
Jensen, C. (2003). QuikNet v2.23, http://qwiknet.home.comcast.net/~qwiknet/.
Levin, R. and C. Kirkpatrick (1966). Planning and Control with PERT/CPM, McGraw-Hill Book Company, New York, NY.
Lippmann, R. P. (1987). "An Introduction to Computing with Neural Nets", *IEEE ASSP Magazine*, Vol. 3, No. 4, April, pp. 4-22.
Meisl, C. J. (1989). "Advanced Life Cycle Cost Modeling", *Journal of Parametrics*, Vol. 9, No. 1, January, pp. 74-102.
Murray, M. and J. Rice (1993). Differential Geometry and Statistics, Chapman and Hall, London.
Rumelhart, D. E, G. E. Hinton, and R. J. Williams (1986). "Learning Representations by Back-Propagating Errors", *Nature*, Vol. 323, October, pp. 533-536.
Smith, M. (1993). Neural Networks for Statistical Modeling, Van Nostrand Reinhold, New York.
Werbos, P, J. (1974). Beyond Regression: New Tools for Prediction and Analysis in The Behavioral Sciences, PhD Dissertation, Harvard University, Cambridge, MA.