A methodology for building neural networks models from empirical engineering data

Yoram Reich
Department of Solid Mechanics, Materials and Systems
Faculty of Engineering
Tel Aviv University
Ramat Aviv 69978
Israel
Email: yoram@eng.tau.ac.il
Tel: +972-3-6407385
Fax: +972-3-6407617

S. V. Barai
Department of Civil Engineering
Indian Institute of Technology
Kharagpur-721 302, West Bengal
India
Email: skbarai@civil.iitkgp.ernet.in.
This research was done while the author was a postdoctoral fellow
at the Faculty of Engineering, Tel Aviv University.

Reference: Reich & Barai (2000), Engineering Applications of AI, 13(5):377-386

Abstract: Neural networks (NN) have become to be general tools for modeling functional relationships in engineering. They are used to model the behavior of products and the properties of processes. Nevertheless, their use is often *ad hoc*. This paper provides a sound basis for using NN as tools for modeling functional relationships implicit in empirical engineering data. First, a clear definition of a modeling task is given, followed by reviewing the theoretical modeling capabilities of NN and NN model estimation. Subsequently, a procedure for using NN in engineering practice is described and illustrated with an example of modeling marine propeller behavior. Particular attention is devoted to better estimation of model quality, insight on the influence of measurements error on model quality, and the use of advanced methods such as stacked generalization and ensemble modeling to further improve model quality. Using a new method of ensemble of $SG(k-NN)$, one could improve the quality of models even if they are close to being optimal.

1 Introduction

The process of building functional models from empirical engineering data is important in all engineering disciplines. Tools for fitting parametric models to data such as non-linear regression require *a priori* selection of a parametric model. This selection introduces bias that might have significant impact on the success of modeling. Non-parametric techniques avoid this problem at the potential cost of additional computational resources. Examples of these methods are k-nearest neighbor (Duda and Hart, 1973), smoothing splines (Wahba, 1990), other non-parametric statistical techniques (Bishop, 1995), and neural networks (NN). NN have gained significant popularity due to two reasons. First, it has been proven that under certain conditions, NN can create nonlinear mappings between input and output variables (Scarselli and Tsoi, 1998). Second, this property has been demonstrated in many applications.

NN popularity is also driven by the wrong belief that using NN for modeling is straightforward or easy. This, in turn, could lead to incorrect use. Two elements are crucial to the application of a modeling method;

- Understanding the theory underlying the modeling method, as well as knowledge of advanced methods for improving modeling results.
- Understanding the process of applying the method to real data including model testing and result interpretation.

This paper addresses these ingredients through ideas and methods that are expected to work well not only on propeller behavior data but also on other similar data. More specifically, the contribution of this paper includes:

- Sections 2 and 3: Reviewing the theory and practice of using NN for data modeling. We formulate the data modeling task and the capabilities of NN models as generalized function estimators and the process of estimating NN models. We focus on the theory and practice of NN usage and demonstrate it on marine propeller behavior data.

- Section 4.1: Discussing the influence of data quality on the quality of models including: measurement errors, model building errors, and human coding errors. We show how NN could be used to improve data integrity (Section 3.2) and analyze the influence of measurement errors on the quality of models. Such analysis requires that the original data measurements and their accuracy be provided.

- Section 4.2: Improving model quality by using advanced modeling techniques such as stacked generalization (SG) (Wolpert, 1992) and ensemble modeling. We combine the two approaches into an ensemble of $SG(k-NN)$ models and show how several good quality models can be combined to further improve model quality.

2 Theory of modeling

Data modeling is a process of building a model of data. We focus on models that are used for predicting values of new input data and not those that are used to better understand the data. We first formulate the data modeling task and subsequently, discuss the theoretical capabilities of NN models and their estimation.

2.1 The abstract modeling task

The problem of building a prediction model from data can be formulated as follows:

> Given a training set, S of size n, $(\mathbf{x}_i, \mathbf{y}_i), i = 1, \ldots, n$, drawn randomly and independently from some unknown joint probability distribution F over the pair $(\mathbf{x}, \mathbf{y})$ of vectors $\mathbf{x}$, $\mathbf{y}$, learn to predict $\mathbf{y}$ from $\mathbf{x}$.

The solution is as follows:

> A modeling tool M uses S to build a model $\hat{f}(\mathbf{x})$ that estimates $\mathbf{y} = f(\mathbf{x})$, i.e., $\hat{\mathbf{y}} = \hat{f}(\mathbf{x})$. Since $\hat{f}$ depends on M and S, we note it by $\hat{f}(\mathbf{x}; S, M)$.

There are always multiple ways to create $\hat{f}(\mathbf{x}; S, M)$. Some methods will have many degrees of freedom and will fit the data perfectly but in doing so will in fact overfit it by wiggling through the data points. The ability of these methods to predict new data will be rather poor. In statistical terms, these models have large variances. Methods that have few degrees of freedom will be smooth, thus have low variance, but might be poor approximations of f and thus be biased. More, formally, the *model bias* reflects the difference between the expected value of the estimation, $\mathcal{E}(\hat{f})$, and the parameter being estimated, f:

$$bias = \mathcal{E}[\hat{f}] - f. \tag{1}$$

The *model variance* σ reflects the variability of the difference between the model and the expected value of the model measured by the standard error σ of the model distribution:

$$\sigma^2 = \mathcal{E}[(\hat{f} - \mathcal{E}[\hat{f}])^2] \tag{2}$$

The best model will be the one with best quality or best accuracy. Intuitively, model selection will be a compromise between the bias and variance of the candidate models. Model quality and this intuition can be formalized. Given S and a particular $\mathbf{x}$, independent of S, the accuracy θ of $\hat{f}$ could be measured by a square loss function:

$$\theta = \mathcal{E}[(\mathbf{y} - \hat{\mathbf{y}})^2] = \mathcal{E}[(\mathbf{y} - \hat{f}(\mathbf{x}; S, M))^2 | \mathbf{x}, S], \tag{3}$$

where $\mathcal{E}[\cdot]$ is the expected value with respect to F.

In general, $\mathcal{E}$ is unknown and therefore, θ could only be estimated; for example, by testing $\hat{f}(\mathbf{x}; S, M)$ on an independent set C of size l, $(\mathbf{x}_i, \mathbf{y}_i), i = 1, \ldots, l$, drawn randomly from the same distribution F. The estimated value of θ will be (for a square loss function):

$$\hat{\theta}(C, S, M) = \frac{1}{l}\sum_{i=1}^{l}(\mathbf{y}_i - \hat{f}(\mathbf{x}_i; S, M))^2. \tag{4}$$

In order to be accurate, l must be very large. Unfortunately, in most real situations we have a fixed size database that must be used for training as well as testing. If the complete data set is used for training and testing (i.e., $S = C$), we obtain the *resubstitution* test which is optimistic. If we divide the data into separate training and testing sets we get the *holdout* test which is pessimistic because not all data is used for training. A better test than these two is the k-fold cross validation (CV). In this test, the data is subdivided into k subsets. The procedure iterates on these subsets and in each iteration, the particular subset is held for testing and the other subsets are used for training. At the end of the process, all the data instances were used once for testing. Although it is a better test, like the former two its results vary with different samplings of the data (i.e., of S and C from F). In addition, the results of both holdout and k-fold CV vary due to data subdivision into training (S) and test (C) sets. If k equals the data size, the test is called *leave-one-out*. For additional details on these tests and others for comparing between models, see Reich and Barai (1997).

Going back to Equation 3, it is easy to show that θ can be decomposed into three terms: bias, variance, plus some noise factor. For a square loss function the decomposition is (Geman et al, 1992):

$$\begin{aligned}
\mathcal{E}[(\mathbf{y} - \hat{f}(\mathbf{x}; S, M))^2|\mathbf{x}, S] &= \mathcal{E}[(\mathbf{y} - \mathcal{E}[\mathbf{y}|\mathbf{x}])^2|\mathbf{x}, S] \qquad \text{"}noise\text{"} \\
&+ (\mathcal{E}_S[\hat{f}(\mathbf{x}; S, M)] - \mathcal{E}[\mathbf{y}|\mathbf{x}])^2 \qquad \text{"}bias\text{"} \\
&+ \mathcal{E}_S[(\hat{f}(\mathbf{x}; S, M) - \mathcal{E}_S[\hat{f}(\mathbf{x}; S, M)])^2] \quad \text{"}variance\text{"},
\end{aligned} \tag{5}$$

where, $\mathcal{E}_S[\cdot]$ represents expectation with respect to the possible S, for fixed sample size n. The "bias" and "variance" terms correspond to those in Equations 1 and 2, respectively. The noise element reflects the variance of $\mathbf{y}$ given $\mathbf{x}$ including measurement and human coding errors.

Typically, there is a tradeoff between the bias and variance terms. The variance can be reduced by smoothing or by modeling data with a restricted number of degrees of freedom (e.g., through the use of parametric techniques). Such constraints on the model would lead to high bias. In contrast, fitting a complicated model (e.g., such as non-parametric models) may lead to low bias but increase the variance. We cannot obtain an *a priori* optimal choice between bias and variance that minimizes the overall error (by manipulating M or S).

For feed-forward neural networks (see next section),varying the number of hidden units effectively can yield models that vary from parametric to non-parametric (Sarle, 1994). Indeed, smoothing can be done by using a network with a small number of hidden units while overfitting can result from a model with many hidden units.

2.2 Theoretical background on NN models and model estimation

Some people state that the use of NN is straightforward as shown in Figure 1. In the first stage of *model estimation*, data is used by the NN to build a model and in the second stage of *model use*, the model is used to predict output of new input data. In contrast to this statement, the use of NN is not straightforward but complex. Often, modeling is done without sound theoretical basis and result interpretations are unsound. Therefore, this subsection reviews the theory of NN modeling subdivided into two separate topics: (1) the model and its capabilities and (2) model estimation.

Figure 1: Put about here

2.2.1 NN models

Briefly, neural networks are computational architectures that combine simple units in an arrangement that can then exhibit complex behavior. One of the most familiar architecture is the feed-forward multilayer perceptron (MLP) or feed-forward multilayer NN. The architecture is composed of layers of units called perceptrons: one layer being the input, another being the output, and in between are additional hidden layers. Figure 2 shows a network with 2 hidden layers.

Figure 2: Put about here

The computation is done as follows. The input to the network - the vector $\mathbf{x}$ - is multiplied by the matrix of weights $\mathbf{W_0}$ and with an additional bias vector $\beta_\mathbf{0}$ yields the input vector to the second layer $\mathbf{W_0 x} + \beta_\mathbf{0}$. The output of units is a function of their input $f_0(\mathbf{W_0 x} + \beta_\mathbf{0})$, where f is called the *activation function*. Those outputs are again input to the second layer; therefore, its output is: $f_1\left(\mathbf{W_1} f_0\left(\mathbf{W_0 x} + \beta_\mathbf{0}\right) + \beta_\mathbf{1}\right)$. The output of the network $\mathbf{y}$ is calculated in a similar manner:

$$\mathbf{y} = f_2\left(\mathbf{W_2}\ f_1\left(\mathbf{W_1}\ f_0\left(\mathbf{W_0}\ \mathbf{x} + \beta_\mathbf{0}\right) + \beta_\mathbf{1}\right) + \beta_\mathbf{2}\right) \tag{6}$$

A common activation function f is the sigmoid function which for output values in the range $[0,1]$ is $f = 1/(1+e^{-x})$.

Many publications deal with the ability of MLP to act as universal approximators of functions. A recent overview mentions some of them including (Scarselli and Tsoi, 1998):

1. Continuous functions on a bounded range (e.g., the n-dimensional cube $[0,1]^n$) could be modeled arbitrarily closely using 1-hidden layer if the number of hidden units is varied as needed (Bishop, 1995). The activation function of the hidden layer must not be a polynomial.

2. Arbitrary functions (e.g., piecewise continuous) could be modeled with 2-hidden layers. These NN may be more efficient for modeling continuous functions (e.g., may require less hidden units or require less model estimation time) than 1-hidden layer NN.

In many cases, theoretical results do not provide guidelines about NN configuration or the number of data instances required to achieve a certain performance. Although, there are studies that provide partial information; for example, a 2-hidden layers NN with a step activation function would require a number of hidden units that is polynomial in the desired error and exponential in the number of inputs (Scarselli and Tsoi, 1998). Other difficulties related to NN models include: (1) the aforementioned results hold for noise-free data, and (2) the models that are used to prove the results might be computationally inefficient. Consequently, we should expect difficulties when applying the theory in practical settings.

2.2.2 NN Model estimation

In order to build an approximation of a function from data the model parameters need to be estimated. Model estimation is performed in the training phase of the MLP. In this phase, the network is presented with a set of input-output pairs $(\mathbf{x}_i, \mathbf{y}_i), i = 1, \ldots, n$. For each input $\mathbf{x}_i$, the network calculates its output $\mathbf{y}_i^o$. Following Equation (4), the sum square error (SSE) for the training set is calculated by:

$$SSE = \sum_{i=1}^{n} (\mathbf{y}_i - \mathbf{y}_i^o)^2. \tag{7}$$

In order to become a good model of the data, SSE must be minimized. This could be done by varying the network parameters, i.e., weights (W) and biases (β). Most often, the minimization is done by gradient descent and the weights are updated using the generalized delta rule. Together, these comprise the *back-propagation* algorithm. Note that if the activation functions f are differentiable (as in the case of the sigmoid function), one can easily derive the gradient of SSE with respect to the weights (e.g. by differentiating SSE in Equation (7) and using Equation (6) to calculate $\mathbf{y}_i^o$). This simplifies the use of the gradient method. Nevertheless, the nature of gradient methods is that they may get stuck in local minima and may not reach the optimal weight. Such problems may be more severe in 2-hidden layer NN but could be addressed by using traditional optimization methods to estimate the model parameters (Sarle, 1994).

There are several theoretical issues that have practical implications in model estimation and model quality assessment:

• **What shall we do if we get additional data?** This question relates to the concept of *consistency* (Geman et al, 1992). A consistent method is one that if given enough data it can estimate any function. With a fixed architecture, NN are inconsistent. In order to make them consistent, their number of hidden units must be increased with the size of the training set. Therefore, additional data would probably require enlarging the NN.

• **What may happen when data changes?** This relates to the concept of *stability* (Breiman, 1996). Briefly, a method is *regularized* if it can be used to build a sequence of models indexed by a sequence of real numbers. For example, MLP with one layer with m hidden units is a sequence of models indexed by m. A regularized sequence is unstable if small changes in the data could cause large changes in the accuracy of models in the sequence. According to this definition, MLP are unstable. Consequently, systematically picking an optimal number of hidden units is difficult because the quality of models may vary significantly when adding or removing units depending on available data. Instability makes model quality assessment hard and model selection even harder since statistical tests for model quality assessment such as leave-one-out or k-fold CV manipulate data in various ways to create their estimates.

Unstable models can be stabilized (Breiman, 1996). This can be done by replacing a single test with the average of several tests, each of which uses the same input-output pairs $(\mathbf{x}_i, \mathbf{y}_i)$ but in each test, noise is created independently from a normal distribution to result different $(\mathbf{x}_i, \mathbf{y}_i + \epsilon_i)$ training sets.

• **Are NN models reliable?** NN in general and MLP in particular replicate known statistical models and extend them in various ways (Sarle, 1994). However, while estimating NN models, no confidence in the estimated parameters is provided. This contrast with confidence intervals associated with estimated parameters of statistical models. Consequently, we cannot know whether particular NN weights are significant or not.

In spite of these difficulties and other issues, NN are promising modeling tools. We shall now elaborate on the practice of modeling.

3 The practical modeling task

3.1 The practice of modeling

The process of modeling real data is more complicated than the theory of model estimation because in contrast to the problem formulation in Section 2.1, many aspects of the practical problem are initially unknown. Only recently the practical use of machine learning (ML) has been recognized as critical by the general ML community (Reich, 1997; Reich, 1998). A seven-step process called *Contextualized ML Modeling (CMLM)* which is shown in Figure 3, can be used for directing modeling.

Figure 3: Put about here

As the figure shows, sometimes these steps must be revisited if subsequent modeling steps fail. For example, better understanding of a problem at the end of step 2 might reveal that critical information was not recorded (e.g., measurement errors, see Section 4.1). Improving the model would then require collecting the missing information. Modeling iterations are costly but are inherent parts of modeling.

There is sufficient evidence in the literature to reveal that none of the seven modeling steps is trivial or could be executed without deep understanding of the issues involved in modeling. To illustrate them, their execution in the context of modeling propeller behavior is detailed next.

3.2 An example: Modeling marine propeller behavior

We now illustrate the seven modeling steps of Figure 3 in the context of modeling marine propeller behavior. Creating behavioral models of marine propeller behavior from empirical test data is extremely valuable for the design iterations of such propellers.

- **Problem Analysis**

The practical modeling task deals with creating a model that can be used to estimate the behavior of a series of propellers at various operating conditions.

The quality of available empirical data is poor and can hardly be used for constructing new models. New good quality data is required (i.e., $(\mathbf{x}_i, \mathbf{y}_i)$, $i = 1, \ldots, n$) that is representative of the possible situations we anticipate to encounter (i.e., S must be representative of C). The data must be sufficient to cover the domain of interest (i.e., enough to "cover" the distribution F); although, collecting large data requires significant resources, especially when tests involve sea trials.

- **Collecting data and knowledge**

The data for our study were created in open sea trials using a modified USN 36 ft. boat fitted with commercial propellers (Denny et al, 1989). The data include 301 instances and cover the following dimensionless parameters: thrust coefficient (K_T), torque coefficient (K_Q), efficiency (η), advance coefficient (J), pitch diameter ratio (P/D), expanded area ratio (EAR), number of blades (Z), and cavitation number (σ). The task is to build a model that maps the input data described by the propeller geometry and operating conditions (i.e., Z, EAR, P/D, J, etc.) to the output which is the performance of the propeller (i.e., K_T, K_Q, and η). The reported accuracy of data collection during the experiment was:

- Propeller rps (n): ± 1 rpm ($\pm 1/1000 = 0.1\%$ full scale; max rpm is not given therefore this figure is an approximation assuming engine rpm to be 1500 and given 1.5:1 reduction gear-box). Note, the full-scale relative error of rps is equal to that of rpm.
- Ship speed (V): ± 0.1 $knot$ ($\pm 0.1/20 = 0.5\%$ full scale; max ship velocity is not given therefore this figure is an approximation).
- Thrust (T): ± 25 lb ($\pm 0.5\%$ full scale; for lower T values, the relative error is larger).
- Torque (Q): ± 64.8 $in - lb$ ($\pm 0.2\%$ full scale; for lower Q values, the relative error is larger).

The experimental data is displayed in many graphs that include all test instances described by the dimensionless parameters. We used data extracted from the original graphical data and given to us by Neocleous and Schizas (1995). In their study, propeller behavior models were created using three types of NN: feed-forward, recurrent nets, and another proprietary net. For each type of NN, different architectures were explored. The accuracy of the models in terms of SSE was evaluated using resubstitution and holdout (248 instances were used for training the NN and 53 test for testing it). These tests are suboptimal; nevertheless, additional, previously conducted tests (Reich and Barai, 1999) and the tests in this study reaffirm that NN can be used to model marine propeller behavior data effectively.

- **Creating representations for the problem, data, and knowledge**

Five parameters - J, P/D, EAR, Z, and σ - were selected as inputs and three - K_T, K_Q, and η - as outputs. Simple normalization was performed to the parameters. These parameters are dimensionless; otherwise, it is theoretically better to transform the data into dimensionless format (Rudolph, 1997).

- **Selecting solution method**

The feed-forward multilayer perceptron was selected due to its recognized ability (Section 2.2) to perform non-linear regression; the apparent smoothness of the function being modeled; and the availability of seemingly sufficient data. We chose to use the implementation (with improved backpropagation) of MATLAB Neural Networks Toolbox (Demuth and Beale, 1994).

- **Selecting neural networks model parameters**

After several training exercises, we fixed the neural network architecture to have two hidden layers with 30 hidden units in each layer. The stopping criteria were SSE equals 0.5 and learning rate equals 0.02, keeping a compromise between the accuracy and the computational time. Note that no optimization of the architecture or training parameters was performed. In contrast, we selected a common architecture and a set of parameters that is suboptimal in terms of quality but that will require reasonable training time. We needed reasonable training time because the statistical exercises conducted in this work require many cycles of training and testing. In situations where no restrictions on computational time are relevant, the setting of parameters should reflect the tradeoff between error bias and variance. Setting these parameters is more an art than a science. They depend on the amount of data and its quality and on the complexity of the function to be approximated.

- **Evaluating and Interpreting Results**

The NN performance was evaluated using several accuracy estimation tests and the results for three instances were found to be poor. They caused errors that ranged between 30% to 170% in the leave-one-out test and also high errors in the resubstitution test. These errors indicate that those instances might be erroneous. Following the inspection, two instances were identified as coding errors introduced in the first extraction of data from the original charts and the third was identified as a suspected data collection error. This activity demonstrates the careful use of tests for checking data integrity or novelty. Subsequent results reported in this paper are based on the original electronic data without any modification.

A good assessment of model quality can be obtained from a 10-fold CV test (see Section 2.1); we performed ten such tests to account for the variability due to the subdivision of data into ten folds. The minimum and maximum values of these runs, as well as the computed mean relative error and standard deviation for the parameters K_T, K_Q, and η are tabulated in Table 1. The sensitivity of estimating K_T to data subdivisions seems larger than the other parameters. The scatter of some of the original graphs (Denny et al, 1989) suggests that we should not anticipate getting more accurate results than the present model provides (see also Section 4.1). Altogether, this exercise demonstrates the non-trivial nature of model quality assessment and its dependence on the quality of data and complexity of function being estimated.

Table 1: Put about here

Note, however, that the results do not include take into account the variability of model quality due to sampling of S and C from F (e.g., the choice of the particular sea trials performed), nor do the variability of model quality due to the choice of learning parameters and NN architecture. The first dependence could be assessed with bootstrapping: another statistical test for model assessment (Reich and Barai, 1999). However, bootstrapping is a tedious and time consuming process. Also, the first dependence has implications to the design of data collection: how should sea trial conditions be selected to maximize model quality? The second dependence could be assessed by a careful study of NN modeling parameters; however, we did not do it in this study.

- **Solution deployment and maintenance**

Our studies show that NN build practical quality models of marine propeller behavior. Our intention is to develop the modeling capabilities further and deploy such models as parts of a complete solution for supporting propeller design (Reich et al, 1997).

4 Understanding and improving model quality

In order to understand and further improve model quality, we focused on two topics:

1. The influence of measurement errors on model quality.
2. Improving modeling using stacked generalization and NN ensembles.

The first topic provides a lower bound on the quality of models we can hope to estimate, and the second topic, generates the best model quality.

4.1 Influence of measurement errors on model quality.

Experimental data always have measurement errors. This raises questions such as: how reliable is the model given that measurement errors exist? The measurement errors in the propeller data collection were given in Section 3.2. Since the modeling uses dimensionless parameters, their errors need to be calculated from raw data. The errors in the five input variables can be calculated as follows:

- P/D, EAR, Z, are assumed to be error-free.
- $J = V_A/nD = V(1-W_T)/nD$; therefore, $\triangle J = J \cdot (\triangle V/V + \triangle(1-W_T)/(1-W_T) + \triangle n/n + \triangle D/D)$.

D is assumed to be accurate. Also, we assume that $(1 - W_T)$ is constant for all operating conditions although it is certainly not so. There are obviously errors associated with calculating W_T as seen from Figure 5 in (Denny et al, 1989). This assumption might contribute significant error that cannot be incorporated into the present analysis. Therefore, based on the available data, a lower bound on the relative error of J is $0.5\% + 0.1\% = 0.6\%$. This figure is a *lower* bound since it sums full scale relative errors that are also *lower* bounds. If we had the original data we could have calculated directly the error of each dimensionless parameter.

- $\sigma = 2gH/V_A^2 = 2gH/(V(1-W_T))^2$; therefore, $\triangle\sigma = \sigma \cdot (\triangle H/H + 2 \cdot (\triangle V/V + \triangle W_T/W_T))$. As before, no error is associated with the calculation of the mean thrust wake factors. The error in measuring H is not given but might be quite significant. Therefore, the *lower* bound of the relative error of J is $2 \cdot 0.5\% = 1\%$ although, in fact, it is much higher.

The errors in the output variables K_T, K_Q, and η are:

- $K_T = T/\rho n^2 D^4$; therefore, $\triangle K_T = K_T \cdot (\triangle T/T + 2 \cdot \triangle n/n)$.
  The lower bound of the relative error of K_T is thus $0.5\% + 2 \cdot 0.1\% = 0.7\%$.

- $K_Q = Q/\rho n^2 D^5$, therefore, $\triangle K_Q = K_Q \cdot (\triangle Q/Q + 2 \cdot \triangle n/n)$.
  The lower bound of the relative error of K_Q is thus $0.2\% + 2 \cdot 0.1\% = 0.4\%$.

- $\eta = (J/2\pi)(K_T/K_Q) = (V(1-W_T)/nD2\pi)(TD/Q)$; therefore, $\triangle\eta = \eta \cdot (\triangle V/V + \triangle n/n + \triangle T/T + \triangle Q/Q)$.
  The lower bound of the relative error of η is thus $0.5\% + 0.1\% + 0.5\% + 0.2\% = 1.3\%$. As before, W_T could have contributed large error to this figure.

These error figures are *underestimated lower bounds* of the true relative errors since: (1) some parameters are assumed to be accurate while in fact, they are not; and (2) the calculated relative errors are correct only for the full scale while for smaller values of parameters, they are higher.

The contribution of measurement errors of output variables to model accuracy is included in the "noise" term in Equation 5. Therefore, even if we minimize the bias and variance, We cannot expect to obtain models with relative accuracy better than the measurement errors or the calculated relative errors as reflected in that term. Moreover, the noise term does not include the contribution due to errors in the input $\mathbf{x}$. Likewise, most studies dealing with noise assume the output parameters are noisy. Input errors may lead to a biased solution that blur or smooth the original function (Tresp et al, 1994).

4.2 Improved modeling by using ensembles

The quality assessment figures obtained by statistical tests such as 10-fold CV are global quality measures averaged over many predictions. They say nothing about the quality of predicting a particular propeller behavior in one particular operating condition. Stacked generalization (SG) is a method that can be used to estimate the local prediction error of another model (Wolpert, 1992). To illustrate (see Figure 4), one algorithm may create a function from $\mathbf{x}$ to $\mathbf{y}$ in the *original data space*, $\hat{\mathbf{y}} = \hat{f}(\mathbf{x})$. The errors between the data and the model prediction are used as input to a second algorithm in the *error space* to create a model between an augmented input $\mathbf{x}'$ and the error $\mathbf{e} = \mathbf{y} - \mathbf{y}^o$, $\hat{\mathbf{e}} = \hat{f}'(\mathbf{x}')$, where $\mathbf{y}$ is the target output from the data set and $\mathbf{y}^o$ is the output of the first model. The augmented input is the original input $\mathbf{x}$ and the description of the instance closest (nearest neighbor) to the new input in the training set. The second model could be used to predict the error that the first model would have when predicting the output of a new input. Together, both models can yield a better estimation that is calculated by $\hat{\mathbf{y}} + \hat{\mathbf{e}}$. To be safe, the contribution of the second model could be halved to yield (Wolpert, 1992):

$$\hat{\mathbf{y}} + 0.5 \cdot \hat{\mathbf{e}} \tag{8}$$

Figure 4: Put about here

Unfortunately, SG does not always work. Figure 5 shows the equivalent of Figure 4 for noisy data. The error in the original space fluctuates around the model and the model of the error is close to zero. SG would fail to improve model quality with such data because the nearest neighbor to a new input does not provide useful information for predicting its output.

Figure 5: Put about here

We are developing improved versions of SG. The original SG uses one nearest neighbor to augment the input space. Our initial improved version uses the k-nearest neighbors instead of one. We denote this version by SG(k-NN). We anticipate improved performance similar to the improved performance when using k-NN regression instead of 1-NN regression in statistics. The best value of k depends on the particular problem and available data.

Many ways exist to create models in the original and error spaces. In our implementation, the models in the original space are the same as before and the model in the error space was implemented by a NN as well. The results of SG(k-NN) for $k = 1, \cdots, 6$, calculated according to Equation 8, are shown in Table 2. The use of the SG(k-NN) algorithm with different k values improves the basic SG and is better than the results of the particular CV test whose data subdivision was used in all the tests (shown as the first entry in the table). Nevertheless, these improvements are small. We repeated these tests several times and obtained similar trends. No statistical tests were conducted because the results are clearly statistically insignificant and conducting such assessment is computationally expensive for SG. From these results we can infer that our situation resembles Figure 5 rather than the one depicted in Figure 4. We should also acknowledge that model quality is very high given the lower bounds derived from measurement errors we calculated before and our understanding that indeed they are *underestimated lower bounds* on model quality. Nevertheless, we can still use the data we have collected thus far to further improve model quality.

Table 2: Put about here

The use of SG(k-NN) has generated a family of models. We could use them as an ensemble of models. Under certain conditions an ensemble performs better than any of its constituent models. In particular, the following heuristics apply: (1) *quality*: the ensemble models should be of good quality; and (2) *diversity*: the errors the models make should not be in the same cases so that they could compensate each other.

Krogh and Vedelsby (1995) showed that an ensemble error can be calculated by:

$$E_j = \overline{E}_j - \overline{A}_j \tag{9}$$

where,
E_j is the ensemble error for jth output parameter,
$\overline{E}_j$ is the weighted average of error of individual networks for jth output parameter, and
$\overline{A}_j$ is the weighted average of ambiguities in jth output parameter.

The weighted average of error of individual networks for jth output parameter, $\overline{E}_j$, is calculated by

$$\overline{E}_j = \sum_{i=1}^{n} w_{ij}\delta_{ij}, \tag{10}$$

and the weighted average of ambiguities in jth output parameter is calculated by

$$\overline{A}_j = \sum_{i=1}^{n} w_{ij}\alpha_{ij}, \tag{11}$$

where,
w_{ij} is the weights assigned to output of jth parameter of ith network model,
δ_{ij} is the SSE of jth parameter of ith network model, and
α_{ij} is the ambiguity in jth parameter of ith network model.

The ambiguity α_{ij} is calculated by

$$\alpha_{ij} = (y^o_{ij} - \overline{y}_j)^2 \tag{12}$$

where,
y^o_{ij} is the output of jth parameter of ith network model, and
$\overline{y}_j$ is the weighed mean output of jth parameter i.e. $\sum\limits_{i=1}^{n} w_{ij}y^o_{ij}$.

Equation 9 separates the generalization error into one term that depends on the generalization errors of the individual models and another term that contains the correlations between them. The first is low if models quality is high (heuristic 1) and the second is low if the diversity is high (heuristic 2). Krogh and Vedelsby (1995) also show how to compute the values of optimal weights

w_{ij}. This calculation requires large data set for training and testing. In cases where a small dataset is available, it is advisable to assign equal weights that lead to a conservative ensemble.

In this study, we assigned equal weights to the different models generated by SG(k-NN). No particular attempt has been made to generate a diverse ensemble; however, the ensemble turned out to be diverse. Meaning, the errors that different models make are related to different predictions. Therefore, the different ensemble models complement each other. The results of the ensemble of SG(k-NN) are given at the end of Table 2. They show significant improvements above each of the models alone and above the original CV test. Again, given the lower bounds on model quality, ensemble modeling moves us very close to the optimal model given the available data and nature of function. Additional details on ensemble modeling can be found elsewhere (Barai and Reich, 1999).

5 Summary and conclusions

In respect to modeling all kinds of problems, no method can be singled out to be superior than their peers (Schaffer, 1994; Wolpert, 1995). In fact, in classification problems, averaged over all modeling problems, all methods are equal. For the modeling of marine propeller behavior, NN are candidates for generating good quality models.

Multilayer perceptron neural networks provide means for modeling arbitrary functions. They can clearly model the behavior of marine propellers. These capabilities were demonstrated with basic NN modeling without any optimization to NN configuration or parameters. New methods for improving the quality of the basic models, SG(k-NN) and ensemble modeling, were discussed and demonstrated. They will work also on other similar problems. These methods generated near-optimal models, given the nature of empirical data available and the function being estimated.

Model building should be done carefully, starting from data collection, model quality estimation, to solution deployment. We presented a seven-step process for modeling data and demonstrated it. Since modeling technology changes rapidly, one has to constantly monitor it. We demonstrated that even in cases when basic modeling performs well, advanced methods can still improve the model quality.

Before closure, one note about research methodology is needed. A great amount of data is collected on a regular basis for various studies in different applications. However, in research reports, some data is manipulated, summarized, or transformed, rendering it unusable for future uses. It will be beneficial for future knowledge advances if this data can be made available in its raw form in central repositories accessible to researchers.

Acknowledgments

Dr. S. V. Barai was supported by a VATAT fellowship. We thank Dr. Costas Neocleous for giving us an electronic version of the data and saving us hours of data decoding.

References

Barai, S. V. and Reich, Y. (1999). "Ensemble modeling or selecting the best model: Many can be better than one." *AI EDAM*, 13(5):377–386.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*, Clarendon, Oxford, UK.

Breiman, L. (1996). "Heuristics of instability and stabilization." *The Annals of Statistics*, 24(6):2350–2383.

Demuth, H. and Beale, M. (1994). *Neual Networks Toolbox - For Use with MATLAB*, The Mathworks Inc., Natick, MA.

Denny, S. B., Puckette, L. T., Hubble, E. N., Smith, S. K., and Najarian, R. F. (1989). "A new usable propeller series." *Marine Technology*, 26(3):173–191.

Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York.

Geman, S., Bienstock, E., and Doursat, R. (1992). "Neural networks and the bias/variance dilemma." *Neural Computation*, 4(1):1–58.

Krogh, A. and Vedelsby, J. (1995). "Neural network ensembles, cross validation, and active learning." In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems, Vol. 7*, pages 231–238, Cambridge, MA, MIT Press.

Neocleous, C. C. and Schizas, C. N. (1995). "Artificial neural networks in marine propeller design." In *Proceedings of ICNN'95 - International Conference on Neural Networks, Vol.2*, pages 1098–1102, New York, NY, IEEE Computer Society Press.

Reich, Y. and Barai, S. V. (1999). "Evaluating machine learning models for engineering problems." *Artificial Intelligence in Engineering*, 13(3):257–272.

Reich, Y., Bertram, V., and Friesch, J. (1997). "The development of a decision support system for propeller design." In *Proceedings of the 9th International Conference on Computer Applications in Shipbuilding (ICCAS '97).*

Reich, Y. (1997). "Machine learning techniques for civil engineering problems." *Microcomputers in Civil Engineering*, 12(4):307–322.

Reich, Y. (1998). "Learning in design: From characterizing dimensions to working systems." *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, 12(2):161–172.

Rudolph, S. (1997). "On topology, size and generalization of non-linear feed-forward neural networks." *Neurocomputing*, 16(1):1–22.

Sarle, W. S. (1994). "Neural networks and statistical models." In *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, pages 1538–1550, Cary, NC, SAS Institute.

Scarselli, F. and Tsoi, A. C. (1998). "Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results." *Neural Networks*, 11(1):15–37.

Schaffer, C. (1994). "A conservation law for generalization performance." In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 259–265, Morgan Kaufmann.

Tresp, V., Ahmad, S., and Neuneier, R. (1994). "Training neural networks with deficient data." In Cowan, J. D., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems, Vol. 6*, pages 128–135, San Mateo, CA, Morgan Kaufmann.

Wahba, G. (1990). *Spline Models for Observational Data*, CBMS-NSF Regional Conference Series in Applied Mathematics; No. 59, SIAM, Philadelphia, PA.

Wolpert, D. H. (1992). "Stacked generalization." *Neural Networks*, 5:241–259.

Wolpert, D. H. (1995). "The relationship between PAC, the statistical physics framework, the Bayesian framework, and the VC framework." In Wolpert, D. H., editor, *The Mathematics of Generalization*, Addison-Wesley.

6 Nomenclature

A	Ensemble ambiguity
C	Testing set
D	Propeller diameter
E, e	Model error
EAR	Expanded area ratio
f	Function from x to y
F	Joint probability distribution
g	Acceleration due to gravity
H	Total static head at shaft centerline
J	Advance coefficient $J = V_A/nD$
K_Q	Torque coefficient, $K_Q = Q/\rho n^2 D^5$
K_T	Thrust coefficient, $K_T = T/\rho n^2 D^4$
l	Size of testing set
M	Modeling tool
n	Propeller rps (revolutions per second)/ size of dataset or training set
P	Propeller pitch
Q	Torque
S	Training set
T	Thrust
V	Ship speed
V_A	Advance speed $V_A = V(1 - W_T)$
$(1 - W_T)$	Thrust wake fraction
W	NN weight matrix
w	Ensemble weights
$\mathbf{x}$	Input vector
y or $\mathbf{y}$	Output or output vector
Z	Number of blades
α	Ensemble ambiguities
β	NN biases
δ	Errors of single ensemble models
η	Propeller efficiency, $\eta = (J/2\pi)(K_T/K_Q)$
ϵ	Noise
$\mathcal{E}$	Expectation
ρ	Water density
σ	Cavitation number / model variance
θ	Model accuracy or error
$\hat{x}$	Estimate of x
$\triangle x$	Measurement or calculated error of x
$\overline{x}$	(Weighted) average of x

Table Captions:

1. Relative error rates for marine propeller behavior in ten 10-fold CV test
2. Error rates of SG(k-NN) and their ensemble

Table 1

	K_T	K_Q	η
Minimum	8.52	4.93	4.34
Maximum	7.48	4.03	3.93
Mean	7.91	4.57	4.18
Std. Dev.	0.36	0.27	0.12

Table 2

Error of Individual Networks			
	Parameters		
	K_T	K_Q	η
10-fold CV $\rightarrow$ No. of neighbors $\downarrow$	**8.09**	**4.47**	**4.20**
1 (original SG)	7.41	4.48	4.30
2	7.03	4.48	4.12
3	7.21	4.57	4.16
4	7.27	4.59	4.10
5	7.33	4.45	4.19
6	7.45	4.56	4.13
Ensemble Error			
$\overline{E}$	7.28	4.52	4.17
$\overline{A}$	1.45	1.03	0.73
E	**5.83**	**3.50**	**3.43**

Figures Captions:

1. Using NN to build mappings
2. MLP architecture
3. CMLM: A model of ML use in practice (Reich, 1997)
4. Error prediction with stacked generalization
5. Error prediction of noisy data with stacked generalization

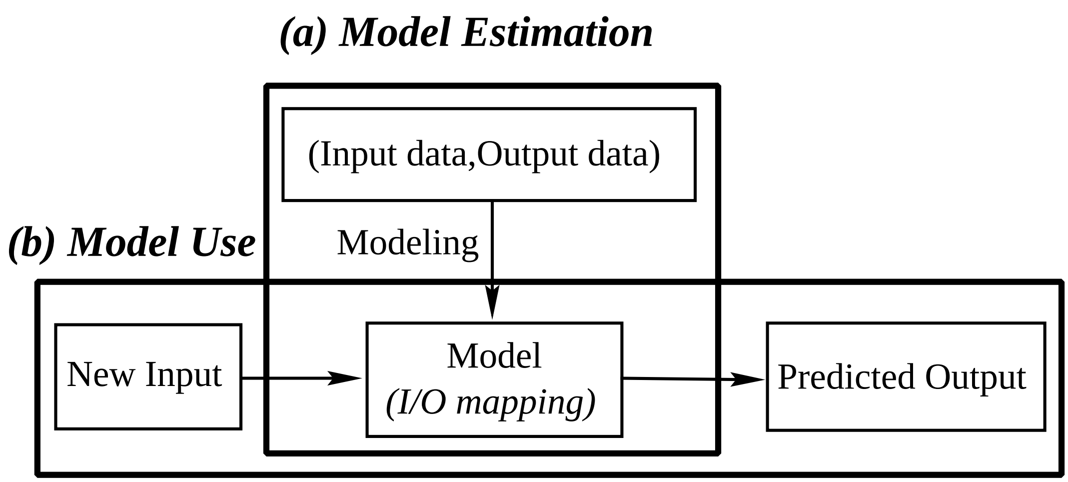
(a) Model Estimation
(Input data,Output data)
Modeling
(b) Model Use
New Input
Model
(I/O mapping)
Predicted Output

Figure 1

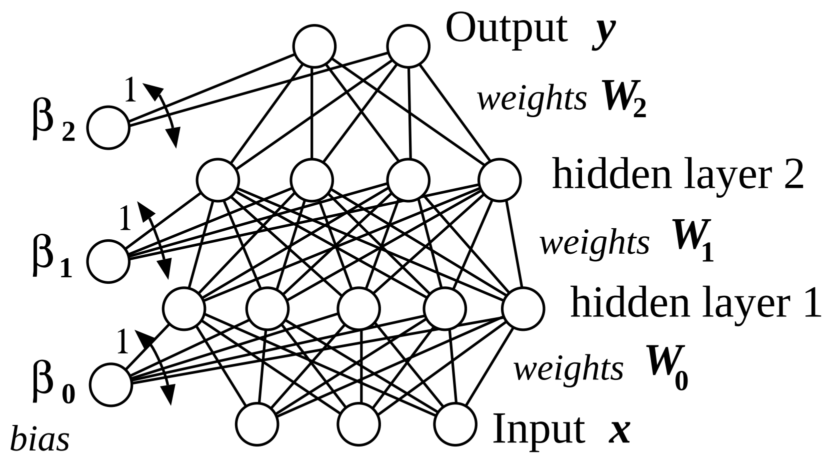
Output y
weights W2
β2
1
hidden layer 2
β1
1
weights W1
hidden layer 1
β0
1
weights W0
bias
Input x

Figure 2

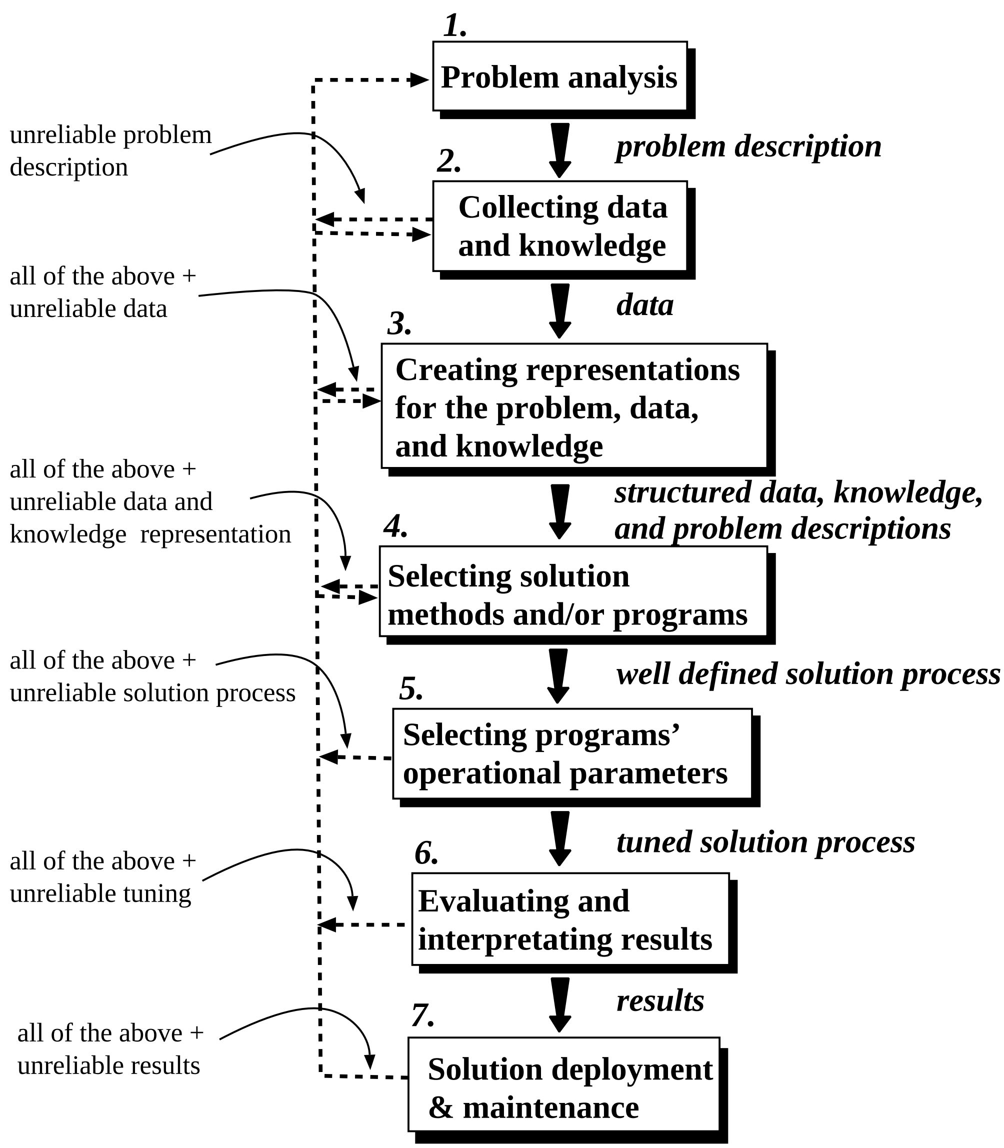
1.
Problem analysis
problem description
2.
Collecting data and knowledge
data
3.
Creating representations for the problem, data, and knowledge
structured data, knowledge, and problem descriptions
4.
Selecting solution methods and/or programs
well defined solution process
5.
Selecting programs' operational parameters
tuned solution process
6.
Evaluating and interpretating results
results
7.
Solution deployment & maintenance
unreliable problem description
all of the above + unreliable data
all of the above + unreliable data and knowledge representation
all of the above + unreliable solution process
all of the above + unreliable tuning
all of the above + unreliable results

Figure 3

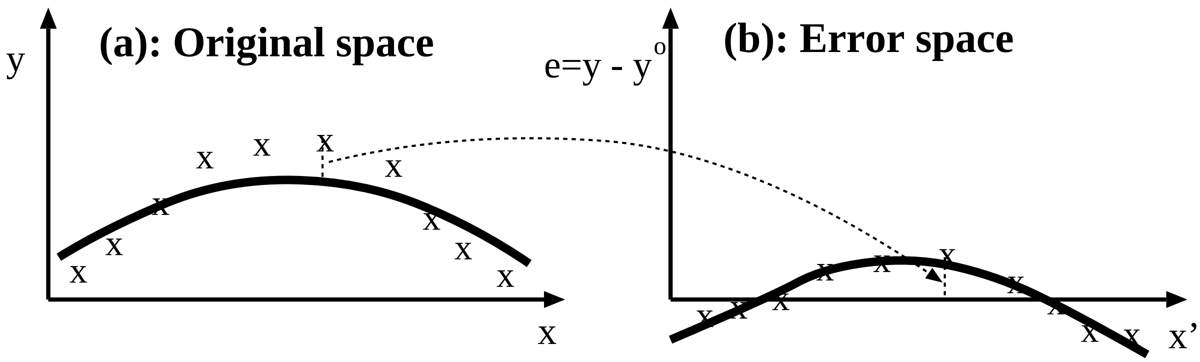

Figure 4

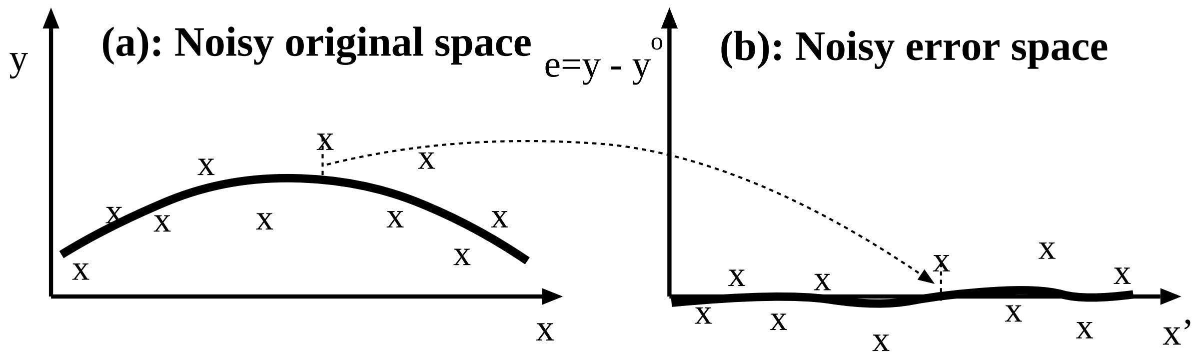

Figure 5