Ensemble modeling or selecting the best model: Many could be better than one

S. V. Barai[1]& Yoram Reich[2]

AI EDAM, in press

Abstract: In the course of data modeling, many models could be created. Much work has been done on formulating guidelines for model selection. However, by and large, these guidelines are conservative or too specific. Instead of using general guidelines, models could be selected for a particular task based on statistical tests. When selecting one model, others are discarded. Instead of losing potential sources of information, models could be combined to yield better performance. We review the basics of model selection and combination and discuss their differences. Two examples of opportunistic and principled combinations are presented. The first demonstrates that mediocre quality models could be combined to yield significantly better performance. The latter is the main contribution of the paper; it describes and illustrates a novel heuristic approach called the *SG(k-NN) ensemble* for the generation of good quality and diverse models that can even improve excellent quality models.

Keywords: ensemble, machine learning, neural networks, data modeling, stacked generalization, model selection

[1]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.

[2]Corresponding author. Department of Solid Mechanics, Materials, and Structures, Faculty of Engineering, Tel Aviv University, Ramat Aviv 69978, Israel. Email: yoram@eng.tau.ac.il

1 Introduction

Given a set of input-output data $(\mathbf{x}, \mathbf{y})$, one can build an unlimited number of models that represent an implicit mapping $\mathbf{y} = f(\mathbf{x})$. There is no best model for arbitrary data or function. In fact, for classification tasks, averaged over all modeling problems, all methods are equal as shown by the *conservation law for generalization performance* (Schaffer, 1994) and the *no-free-lunch (NFL) theorem* (Wolpert, 1995).

In the traditional approach, the information available on the problem is examined and a choice is made among available models that might best solve the problem. Such choice could employ heuristics or guidelines derived from experience. The choice might fail because the true function f is unknown, the information available is limited or even erroneous, and the guidelines or heuristics would be overgeneralization of past experiences.

Inevitably, through the modeling process, the data nature is better understood and new insight about the modeling problem emerges. Consequently, the process iterates where different models are examined or additional data is collected (Reich, 1997; Reich, 1998). At the end of modeling, the intermediate models are usually discarded.

A more systematic model selection will follow a comparative statistical testing among candidate models. Such testing would require large data for model building and tuning, and for model comparison. A common test to use is cross validation but others, especially for classification, are explored in the machine learning (ML) community (Reich and Barai, 1998a).

Whether a model is selected systematically or not, it is clear that the remaining models that were developed and discarded contain potentially valuable information about the problem. If we wish to better understand the nature of data and the process that generated it, and not necessarily estimate the function f for making future predictions, we could definitely benefit from multiple perspectives provided for by different models (Reich et al, 1996).

It has become clear that prediction models could also be improved by the combination of multiple models into one, an approach called *ensemble modeling* (see Sharkey, (1996) for a summary). Numerous practical modeling problems were solved successfully using ensembles. However, we must not think that the combined model or ensemble is guaranteed to perform better than any of its constituents or that the same combination approach will work equally well on another arbitrary modeling problem. This is a corollary of the conservation law or NFL theorem. The choices between models are merely deferred to a meta level: which models to select as the ensemble members and how to combine them?

An important concept that underlies model selection or combination is the *bias-variance tradeoff* (Geman et al, 1992). These could be explained in terms of this tradeoff. This tradeoff emerges from decomposing the expected value of the error of an estimator $\hat{f}$ of some function y into three terms: bias, variance, plus some noise factor. For a square loss function the decomposition of the error can be written very generally as (for a precise formulation see (Geman et al, 1992)):

$$\begin{aligned} \mathcal{E}[(y-\hat{f}(x))^2] &= \mathcal{E}[(y-\mathcal{E}[y])^2] \qquad \text{``}noise\text{''} \\ &+ (\mathcal{E}[\hat{f}(x)]-\mathcal{E}[y|x])^2 \qquad \text{``}bias\text{''} \\ &+ \mathcal{E}[(\hat{f}(x)-\mathcal{E}[\hat{f}(x)])^2] \quad \text{``}variance\text{''}, \end{aligned} \tag{1}$$

where, $\mathcal{E}[\cdot]$ represents expected value. The noise element reflects the variance of y given x including measurement and human coding errors; it is fixed for the data. It turns out that when model complexity increases the bias is reduced but the variance is increased. Therefore, there is a tradeoff between the two terms that yields the "optimal" estimator. This insight can be used to design better estimators or select between them.

This paper reviews the problem of model selection and model design guidelines in the context of neural networks modeling (Section 2). We discuss heuristics for combining models into an ensemble in order to improve performance and review a formalization that makes these heuristics explicit (Section 3). We present two examples of ensemble modeling (Section 4); one example demonstrates that opportunistic ensemble modeling based on models generated in an exploratory iterative modeling process can lead to improved performance even when data quality is poor. This example highlights the problems of models generation and combination which are addressed by the second example. The second example constitutes the main contribution of this paper; it describes and demonstrates a new method based on a set of modified stacked generalization instantiations (Wolpert, 1992) for systematically generating quality ensembles. The method is called the *SG(k-*

NN) ensemble. This method succeeds in improving results even when it seems that close to optimal performance has already been achieved.

2 Model selection

Model selection is one of the steps in building models from data (Reich, 1997; Reich, 1998). It has been perceived that using the best single tool for solving a problem is the best solution approach. However, it is also recognized that no ML tool is better than all others on all problems (Schaffer, 1994; Wolpert, 1995). A critical issue is therefore determining for each model the class of problems it can solve best. This requires collecting data on the use of different tools for different problems including successes and failures and compiling such a mapping (Reich, 1994). Such compilation requires significant effort. An attempt to use ML to assist in this task is demonstrated in the StatLog project. Following the testing of 22 classification programs on more than 20 databases, knowledge was extracted in the form of heuristics that can direct ML program selection given a particular learning problem (Michie et al, 1994). It is unclear whether this approach can be generalized to handle the variety of learning situations and solution methods.

Depending on the choice of modeling approach, several parameters are often available for selection. For example, in the context of modeling a function by neural networks (NN), the following choices are available:

- *Model type.* This selection is based on the particular problem: A simple feedforward network might be sufficient for modeling a simple function, while a recurrent network might be better for a time-dependent function. In the former case, both multilayer perceptron (MLP) network or a radial basis function (RBF) network could be used.
- *Model configuration or topology.* This choice is one of the most common design decisions: determining the number of hidden layers, the number of hidden units, and the type of activation functions of the NN model. Model complexity must produce a good tradeoff between the *bias* and *variance* of the model error (Geman et al, 1992).
- *Model estimation.* First, the error or cost function between the training data and NN output

needs to be formulated. Second, there are two broad categories of model estimation methods to minimize this function. The first method is the one developed for the particular NN such as back propagation for a MLP. The second method involves using minimization programs to minimize the error or cost function directly (Sarle, 1994).

- *Implementation.* Many implementations of NN algorithms are available. They vary in reliability and details. For example, the generation of random numbers may be different thus leading to different initializations of NN weights. Such discrepancies undermine the ability of different implementations to replicate results (Nabney et al, 1997).

There are two methods for generating guidelines: theoretical and empirical. To illustrate them, consider the choice of model configuration.

Theoretical guidelines.

Many guidelines are based on interpretations of theoretical results related to NN. In particular, there are various proofs that NN are universal approximators. For example, Kolmogorov's *mapping neural network existence theorem* (Hecht-Nielsen, 1990) states that every real continuous function $f : [0,1]^d \rightarrow R$ can be written as

$$y_i = \sum_{h=1}^{2d+1} \phi_i \left(\sum_{k=1}^{d} \lambda^h \psi\left(x_k + h\epsilon\right) + h \right) \tag{2}$$

where λ is real, ψ are continuous real monotonically increasing functions independent of f, ϵ is a rational number as small as desired, and ϕ_h are continuous real functions dependent on f and ϵ. Some authors interpreted this as a guideline that in no situation does one need more hidden units h than $2d+1$, where d is the number of inputs (Swingler, 1996). This, however, is wrong since the activation functions required to prove the theorem are problem dependent, unknown, and certainly do not resemble those used in any common NN architecture.

Another related theorem states that a two-hidden layer 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 (Kůrková, 1991; Scarselli and Tsoi, 1998). Consequently, $h = O(a^d)$, for some constant a. The proof of this theorem is constructive: it describes the method of building the NN and its usage. Obviously, the two results are markedly different. If we use any of them as guidelines for one or two-hidden layer MLP with sigmoid activation function, we would have abstracted those results out of their context, rendering them useless. Swingler (1996) mentions many such guidelines with their origin in theoretical results. However, he wrongly maintains the bound on hidden units as $h \leq 2d + 1$.

Theoretical guidelines can also emerge from different theoretical analyzes. For example, Fu and Chen (1993) suggested using $b \leq 4$ in the sigmoid function $1/(1 + e^{-bx})$ of an MLP instead of $b = 1$ in order to reduce the sensitivity of the MLP output to variations in inputs.

Many other theory-based guidelines are based on NN degrees of freedom or various criteria such as the Vapnik-Chervonenkis (VC) dimension (Lawrence et al, 1996). In one such example, Baum and Haussler (1989) relate the size of a MLP with linear threshold functions, the number of training examples, and the training error to the confidence in future predictions. However, the bounds derived from those analyses are too conservative. There is still a gap between theoretical and empirical results in NN as the gap associated with symbolic ML (Turney, 1991).

Empirical guidelines.

Most often, empirical guidelines are based on parametric studies involving various NN architectures applied to several databases or modeling problems. The guidelines are then generalized from the studies (Carpenter and Hoffman, 1997). In performing such tests, attention should be given to the following issues. Similar to what we know about ML, in order to get useful generalized guidelines they have to be extracted from modeling problems that are representative of the problems we might encounter in the future. The number of these problems must be sufficient to obtain meaningful generalization.

The generalized guidelines should be formulated carefully following analyzing the results with acceptable statistical tests. In particular, when comparing multiple models on multiple modeling problems, the problem of multiplicity must be addressed so as to minimize spurious statistical results (Feelders and Verkooijen, 1995; Reich and Barai, 1998a).

In addition, statistical tests have their own bias and variance. Some tests (e.g., cross validation), are more accurate (i.e., have less bias) than others but must be interpreted with care. It is best to control their variance with respect to test execution conditions and data sampling by performing several tests and averaging the results. One such test is K^I, where I runs of k-fold cross validation (CV) are averaged (Reich and Barai, 1998a).

A k-fold CV is performed as follows (Reich, 1997). The data D is divided into k subsets of roughly equal size. The ML program is trained k times, each time leaving out one of the subsets from training, and using it for testing. The error estimation is the average accuracy of the k runs. If $k = n$, n ths size of the data set, the test is called a *LOO* test. It has been common in general ML studies to use a 10-fold CV method when the number of instances, n, exceeds 100, or a leave-one-out method for small databases.

Instead of formulating general guidelines, one can employ statistical tests to select a particular model for a particular modeling problem. The process is similar to the generation of general guidelines except that only one modeling problem is solved and less testing is performed since the selection need not apply in general.

3 Ensemble modeling

No guideline is always correct. No single method is always the best. This has lead to the idea of trying to combine models into an ensemble rather than selecting among them. The idea seems to work well as demonstrated by many practical classification applications (although note that failures are rarely reported). Wolpert (1992) proposed two heuristics for assessing the potential of an ensemble.

1. The ensemble should span the space of generalizers. The ensemble members should be of different types and not merely variants on the same model type.

2. The ensemble models should be "orthogonal". This ensures that each adds additional information towards building an accurate model. Correlated models are not useful for ensemble construction.

These heuristics are clear when using ML to gain better understanding of data: the use of diverse methods provides different perspectives, and many different perspectives improve understanding

(Reich et al, 1996). Krogh and Vedelsby (1995) formalized these heuristics as an instance of the bias-variance tradeoff (Geman et al, 1992). The ensemble error is given by:

$$E_j = \overline{E}_j - \overline{A}_j \tag{3}$$

where,

$\overline{E}_j = \sum_{i=1}^{n} w_{ij}\delta_{ij}$ is the weighted average of error of individual networks of the jth output parameter;

$\overline{A}_j = \sum_{i=1}^{n} w_{ij}\alpha_{ij}$, is the weighted average of ambiguities (α_{ij}, defined below) in the jth output parameter;

w_{ij} is the weight assigned to the output of the jth parameter of the ith model; these weights satisfy $\sum_i w_{ij} = 1$;

δ_{ij} is the sum square error (SSE) of the jth parameter of the ith model;

$\alpha_{ij} = (y_{ij} - \overline{y}_j)^2$ is referred to as the *ambiguity* in the jth parameter of the ith model;

y_{ij} is the output of the jth parameter of the ith model; and

$\overline{y}_j$ is the weighted average output of the jth parameter, i.e., $\sum\limits_{i=1}^{n} w_{ij}y_{ij}$.

Equation 3 separates the generalization error into one term that depends on the errors of the individual models and another term that contains all the correlation between them. The first is low if models quality is high (heuristic 1) and the second is high if the diversity is high (heuristic 2). Krogh and Vedelsby (1995) also show how to compute the values of optimal weights w_{ij}. However, this calculation requires a large data set for training and testing. Thus, in cases where a small dataset is available, it is advisable to assign equal weights that lead to a conservative ensemble.

Figure 1 shows the differences between the model selection and combination approaches. The general framework is similar however, the combination approach has many more degrees of freedom. It also has many more opportunities to address the bias-variance tradeoff. Instead of optimizing the tradeoff for each model before combination, one can generate basic models with lower bias and let the ambiguity of the ensemble (i.e., the ambiguity term in equation 3) reduce the variance (Naftaly et al, 1997). This requires integrating tightly the two steps in the model combination approach.

Put Figure 1 about here

Figure 1: Model selection versus combination

Most work on ensembles to date have dealt with classification. We will present two examples of using ensembles for multidimensional regression.

Lawrence et al, (1996) pointed out that the more noisy the data, the more beneficial is ensemble construction. Our first example has two levels of noise. Our results lead to similar findings: the advantage of using ensemble increases with the level of noise in, or variance of, the data. We also show that ensembles do not always improve results.

Most previous work on ensembles have not dealt with actively generating a good set of diverse models. Opitz and Shavlik (1996) operationalized Equation 3 into a procedure for such generation. Their algorithm is based on genetic search in the space of NN configurations for those NN that contribute most to lowering the E_j's. Our second example implements a novel heuristic approach for systematic generation of good quality ensemble models.

4 Opportunistic and principled ensemble modeling: case studies

We illustrate two approaches of ensemble modeling: *opportunistic* and *principled*. The opportunistic approach emerges from the usual iterative data modeling process. During modeling, various models are explored, the problem is gradually better understood, and hopefully, modeling improves. Yet, at the end of the process, one remains with all the intermediate models, some of which may perform better than the final model. Instead of discarding these models, ensemble modeling combines them into one. If the models are quite different (i.e., diverse) and reasonably good, the ensemble will improve upon the best of them. We demonstrate this approach through modeling corrosion data.

The principled approach seeks to generate systematically a set of as accurate as possible and diverse models from which a single model is composed. We develop a new method with these properties and demonstrate it through modeling marine propeller's behavior data.

4.1 Opportunistic method: Combining intermediate models

This exercise deals with the stress corrosion cracking (SCC) of a sensitized, wrought type 304 stainless steel (Congleton et al, 1995). The goal of modeling is establishing relationships between the environmental conditions and their effects on the steel. The environmental conditions are temperature (T), potential (V), solution types (ST1 and ST2) and the effects are crack length (CL), ultimate tensile strength (UTS), time of failure (TF), reduction of area (RA), and the crack type (CT1 with four distinct values or CT2 with two aggregated values "yes" and "no-crack"). Preliminary data analysis suggested that the data, consisting of 93 instances, is sparse, noisy, and its quality is rather poor.

Neural Networks Models

We selected multilayer perceptron (MLP) and a self-organizing map (SOM) for creating input-output mappings and for locating possible outliers in the data. We used the implementations in the MATLAB Neural Network Toolbox (Demuth and Beale, 1994). From these two basic models, five model combinations were synthesized and summarized in Table 1.

- Model 1: This is the basic model where MLP creates one mapping between the input and output parameters.
- Model 2: This modeling is based on the assumption that classifying the data points into similarity regions will improve predictability. First, SOM forms clusters considering all the parameters. Second, a model is built to recognize these clusters given input data only. Third, a model is built to map the input parameters and the class onto the output parameters.
- Model 3: This approach tries to subdivide the prediction task into two steps. First, the CT1 is predicted and second, CT1 and the other input parameters determine the other output parameters.
- Model 4: Same as model 1 except that *whitening* (Bishop, 1995) is performed on the data before training. Whitening is a linear transformation with correlations of attributes that is done using eigenvectors calculated from the data.

- Model 5: Same as model 1 except that CT1 is replaced by CT2.

Put Table 1 about here

Selection of Neural Networks Model Parameters

The topology and training parameters (number of epochs, and learning rate (lr)) of the different models are given Table 1. The topology specifies the number of input units, units in two hidden layers, and number of output units. SSE was set to 0.005, but was never reached in our experiments; rather, the training cut-off was determined by the number of epochs. There was no optimization of the parameters. Rather, we selected parameters that gave reasonable results with reasonable execution time to allow us to execute the study.

Evaluating and Interpreting Results

Due to the small data size we used *leave-one-out* (*LOO*) to determine the accuracy of the NN models (Reich, 1997; Reich and Barai, 1998a). Two cases were analyzed: the first case used the *raw data* and the second case used *cleaned data*. The data cleaning was carried out manually considering the following criteria:

- Focusing on the parameter RA and removing inconsistent or repetitive examples from the data set.
- Performing resubstitution and LOO tests on the raw data. Identifying patterns that gave high errors for the output parameters and removing them from the data.

In the context of RA, 14 patterns were removed from original data set.

Put Table 2 about here

Put Table 3 about here

The results of the LOO tests in terms of SSE for each model and the ensemble results for the two cases are shown in Tables 2 and 3. In the case of raw data (Table 2), the ensemble improved the predictive accuracy of all four parameters by a significant amount, in particular the accuracy of the two difficult to predict parameters: CL and RA. In the case of cleaned data (Table 3), we expected less improvement. Indeed, this was the case and for one parameter (TF), the ensemble performed worse than the best model.

We did not perform any statistical test to assess the significance of the results we obtained. Such tests would be extremely time consuming. However, in order to study the sensitivity of the ensemble to model's availability, we calculated the ensemble accuracy for each combination of the six models. The maximum and minimum errors, with the model combinations that generated them, are shown at the end of Tables 2 and 3. The results show significant variations. Two of the combinations in Table 3 (for parameters UTS and Time to failure) lead to ensembles that are worse than the best model. Consequently, ensembles are not guaranteed to improve results upon the best model. It becomes critical to determine a systematic method for generating diverse, good quality ensembles.

4.2 Principled method: The *SG(k-NN) ensemble*

The principled method involves using stacked generalization (SG) in an innovative manner. SG is a method for improving the accuracy of one model or combining several models into an ensemble (Wolpert, 1992). To improve one model, SG is used in the following manner (see Figure 2). One model may represent 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 dataset 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 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):

$$\hat{\mathbf{y}} + 0.5 \cdot \hat{\mathbf{e}} \qquad (4)$$

Put Figure 2 about here

Figure 2: Error prediction with stacked generalization

The process of applying SG to a model extracted from a data set is quite involved. Figure 3 shows the data arrangement for SG and Figure 4 provides the algorithm. The figure shows one iteration of testing SG on T after building it from D. If D is composed of $k-1$ folds of a CV test and T is the last fold, the procedure can iterate k times to yield a complete k-fold CV test.

Put Figure 3 about here

Figure 3: Stacked generalization data management

Put Figure 4 about here

Figure 4: The stacked generalization algorithm

The original SG uses one nearest neighbor to augment the input space. Instead of using one nearest neighbor, we employ k-nearest neighbors. We anticipate improved performance similar to the improved performance when using k-NN regression instead of 1-NN regression in statistics. Thus by varying k we can generate different models that are denoted by SG(k-NN), $k = 1, \ldots, n$. For each problem and a database, there is an optimal value for k that will yield the best performance. k also depends on the level of noise (Lawrence et al, 1996). We experimented with $n = 6$. This exercise yields good quality and evidently, quite diverse models.

With the same approach we could have used SG to combine models instead of improve one. The same scheme would then become an ensemble of ensembles.

Case Study Problem Definition

This study deals with predicting the behavior of marine propeller given certain operating conditions and design parameters. The data were created in open sea trials (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).

Neural Networks Models

An MLP was selected due to its proven ability 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).

Selection of Neural Networks Model Parameters

Three NN were used in this study as shown in Figure 4: ATM, LTM, and EPM. Their topology and parameters were:

1. ATM: 5-30-30-3, SSE=0.5, lr=0.02; SSE was the governing stopping training criterion.
2. LTM: 5-30-30-3, SSE=0.5, lr=0.02; SSE was the governing stopping training criterion.
3. EPM: 5-30-30-3, SSE=0.05, epochs 50,000; number of epochs was the governing stopping training criterion.

As before, there was no optimization of parameters involved. Again, we selected parameters that gave reasonable results with reasonable execution time to allow us to perform the study.

The set of models we obtained was then used to create an ensemble by assigning equal weights to the different SG(k-NN) models.

Evaluating and Interpreting Results

A basic 10-fold CV test was performed whose data subdivisions were used in all other tests. SG(k-NN), $k = 1, \cdots, 6$, were created and their results calculated according to Equation 4. Further, the ensemble approach was used on the SG(k-NN) results according to 3. The results of these exercises are tabulated in Table 4. The use of the SG(k-NN) algorithm with different k values improved the basic SG and was 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 rather small. In contrast, the ensemble results show significant improvement above each of the models alone and above the original CV test.

The ensemble of six SG(k-NN) models gave results better than any other sequence with lower k. However, different combinations of different models gave better (or worse) results for the different parameters. The maximal and minimal accuracies and the models that participated in these ensembles are given at the end of Table 4.

Interestingly, the worst results are associated with an ensemble of the first two models. This could be explained by the high correlation that exist between them. In contrast, the best results involve ensembles composed of either the 4th and 5th SG models or the 5th and 6th models. Instead of the optimal value of k that we find in k-NN, there is a different k value that optimizes the ensemble accuracy. We intend to explore whether this observation applies to other situations.

Note on statistical tests.

As noted earlier, no statistical tests were performed. Conducting such tests with SG is extremely time consuming. For example, the computational cost of obtaining the values in Table 4 could be decomposed as follows (see Figures 3 and 4). Testing the accuracy of one SG is done with k-fold CV. In each of the k iterations, an SG is developed from $k-1$ subsets of the total k (the set D in Figure 3). This development is done with CV that is executed in an inner loop. Consequently, in the process of testing one SG, k^2 models are trained from $n \cdot ((k-1)/k)^2$ instances, and $2k$ models are trained with $n \cdot (k-1)/k$ instances, where n is the size of the data. For six models with $k = 10$ and $n = 301$, these numbers are 720 training sessions with a database of about 250 instances. Any statistical test would involve at least 10 executions of a similar procedure—a very costly endeavor. Instead of statistical tests, we provide a different measure of the quality of these results: Elsewhere (Reich and Barai, 1998b), we conducted an analysis of the measurement error in the collection of data for this problem and found that our ensemble results come close to being optimal considering these errors. Additional confidence in this technique will emerge from future applications.

Put Table 4 about here

5 Conclusions

Modeling data is a hard problem that often requires a long iterative modeling process where different choices are explored and evaluated. Most often a model would include useful information even if it is inferior to all other models. Even opportunistic combination of models generated in the course of iterative modeling could be useful. We demonstrated this by combining five models created in the course of modeling material corrosion data.

Engineering data, particularly those generated in experimental setting are often noisy, sparse, and of mediocre quality from the perspective of NN modeling. Fortunately, the usefulness of using ensembles increases when the data quality is poor, since the ensemble averaging reduces the variance contribution to model error. Ensembles can also succeed if data quality is good and the basic models are selected and combined carefully. Nevertheless, there are also chances that the combination of models will result in models less accurate than the best model available.

All these phenomena were demonstrated in the exercise of modeling material corrosion; they suggest that additional work is needed on the generation and combination of good, diverse models and on the influence of the general goal of modeling—obtaining accurate ensemble—on the training of the basic models.

We presented a novel heuristic approach called the *SG(k-NN) ensemble* to the systematic generation of good quality and diverse ensembles. The approach was tested on good quality data and proved useful in improving the best single basic model we generated. Even the worst combination performed better that any single model. This exercise demonstrates that careful generation of ensembles can improve on good-quality models created from good-quality data.

In spite of these results, no general method will work always. Therefore, we can only state that a particular method for creating an ensemble can be better than the best single model and continue to work on identifying the generation and combination methods that can best solve different classes of data modeling problems.

References

Baum, E. B. and Haussler, D. (1989). "What size net gives valid generalization?." *Neural Computation*, 1(1):151–160.

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

Carpenter, W. C. and Hoffman, M. E. (1997). "Guidelines for the selection of network architecture." *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, 11:395–408.

Congleton, J., Berrisford, R., and Yang, W. (1995). "Stress corrosion cracking of sensitized type 304 stainless steel in doped high-temperature water." *Corrosion Science*, 51(12):901–910.

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.

Feelders, A. and Verkooijen, W. (1995). "Which method learns most from the data?." In *Preliminary papers of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 219–225.

Fu, L. and Chen, T. (1993). "Sensitivity analysis for input vector in multilayer feedforward neural networks." In *Proceedings of 1993 IEEE International Conference on Neural Networks (ICNN '93)*, pages 215–218, New York, NY, IEEE.

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

Hecht-Nielsen, R. (1990). *Neurocomputing*, Addison-Wesley, Reading, MA.

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

Kůrková, V. (1991). "Kolmogorov's theorem is relevant." *Neural Computation*, 3(4):617–622.

Lawrence, S., Giles, C. L., and Tsoi, A. (1996). "What size neural network gives optimal generalization? Convergence properties of backpropagation." Technical Report UMIACS-TR-96-22 and CS-TR-3617, Institute for Advanced Computer Studies, University of Maryland, College Park MD 20742, April.

Michie, D., Spiegelhalter, D., and Taylor, C. C. (1994). *Machine Learning, Neural and Statistical Classification*, Ellis Horwood Publishers, Chichester, England.

Nabney, I. T., Paven, M. J. S., Eldridge, R. C., and Lee, C. (1997). "Practical assessment of neural network applications." In Daniel, P., editor, *SafeComp 97, Proceedings of the 16th International Conference on Computer Safety, Reliability and Security*, pages 357–368, Berlin, Springer-Verlag.

Naftaly, U., Intrator, N., and Horn, D. (1997). "Optimal ensemble averaging of neural networks." *Network: Computation in Neural Systems*, 8(3):283–296.

Opitz, D. W. and Shavlik, J. W. (1996). "Generatring accurate and diverse members of a neural-network ensemble." In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems, 8*, Cambridge, MA, MIT Press.

Reich, Y. and Barai, S. V. (1998). "Evaluating machine learning models for engineering problems." *Artificial Intelligence in Engineering*, (in press).

Reich, Y. and Barai, S. V. (1998). "Modeling marine propeller behavior with neural networks.". Submitted to publication.

Reich, Y., Medina, M., Shieh, T.-Y., and Jacobs, T. (1996). "Modeling and debugging engineering decision procedures with machine learning." *Journal of Computing in Civil Engineering*, 10(2):157–166.

Reich, Y. (1994). "Macro and micro perspectives of multistrategy learning." In Michalski, R. S. and Tecuci, G., editors, *Machine Learning: A Multistrategy Approach, Vol. IV*, pages 379–401, San Francisco, CA, Morgan Kaufmann.

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.

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.

Sharkey, A. J. C. (1996). "On combining artificial neural nets." *Connection Science*, 8(3-4):299–313.

Swingler, K. (1996). *Applying Neural Networks: A Practical Guide*, Academic Press, London, UK.

Turney, P. (1991). "The gap between abstract and concrete results in machine learning." *Journal of Experimental and Theoretical Artificial Intelligence*, 3(3):179–190.

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.

Figure Captions

Table Captions

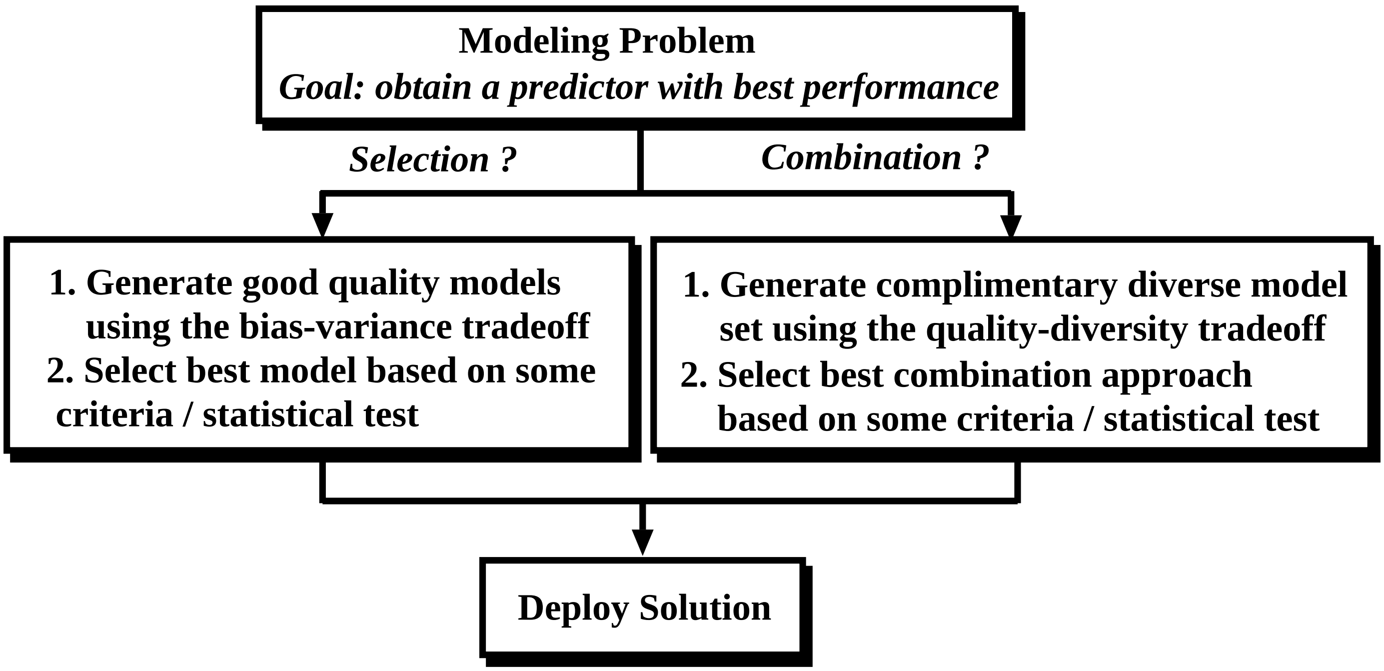
Modeling Problem
Goal: obtain a predictor with best performance
Selection ?
Combination ?
1. Generate good quality models using the bias-variance tradeoff
2. Select best model based on some criteria / statistical test
1. Generate complimentary diverse model set using the quality-diversity tradeoff
2. Select best combination approach based on some criteria / statistical test
Deploy Solution

Figure 1

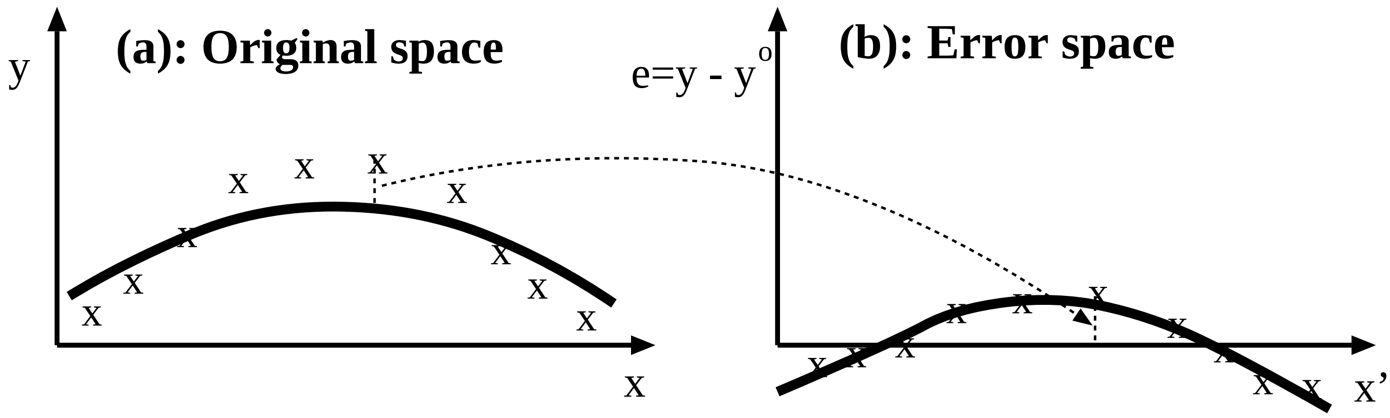

Figure 2

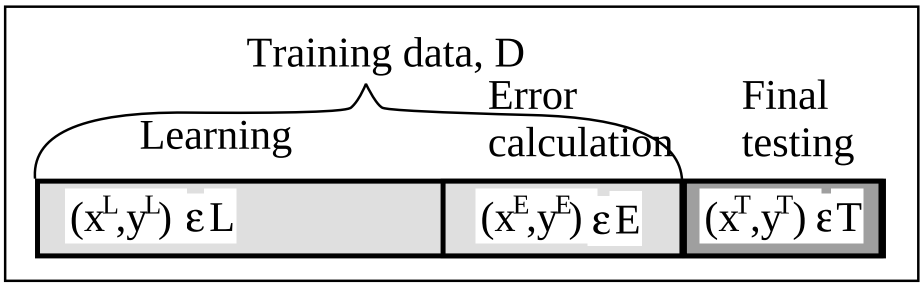
Training data, D
Learning
Error
calculation
Final
testing
(x^L,y^L) ε L
(x^E,y^E) ε E
(x^T,y^T) ε T

Figure 3

Training

1 Create the prediction model from all training data D (ATM)

ATM: $x_i^D \rightarrow y_i^D$

2 Repeat on all subdivisions of D to L and E ($L \cup E = D$)

LTM: $x_i^L \rightarrow y_i^L$, $e_i = (y_i^E) - (y_i^E)^o$, (y_i^E) is the target value, $(y_i^E)^o = LTM(x_i^E)$ is the output value.

This repetition creates the complete set of e_i^D.

3 Create the error prediction model EPM

EPM: $\begin{pmatrix} x_i^D \\ NN(x_i^D) \end{pmatrix} \rightarrow e_i^D$, $NN(x_i^D)$ is the nearest neighbor of x_i in D

Testing

4 Test on all items in T

$EPM(x_j^T, NN(x_j^T)) = e_j$

$ATM(x_j^T) = (y_j^T)^o$

$y_{SG} = (y_j^T)^o + e_j$, $e_{SG} = e_{ATM} - e_j$

Figure 4

Table 1: A summary of modeling approaches

Model	Step	**Input Parameters**	**Output Parameters**	**Modeller**	**Topology**	**Epochs**	**lr**
1	1	ST1, ST2, T, V	CL, UTS, TF, RA, CT1	MLP	4-18-18-6	25,000	0.02
2	1	ST1, ST2, T, V	Classification	SOM [*]	———	2,000	0.1
	2	ST1, ST2, T, V	Class	MLP	4-18-18-1	10,000	0.02
	3	ST1, ST2, T, V, Class	CL, UTS, TF, RA, CT1	MLP	5-18-18-6	25,000	0.02
3	1	ST1, ST2, T, V	CT1	MLP	4-18-18-2	25,000	0.02
	2	ST1, ST2, T, V, CT1	CL, UTS, TF, RA	MLP	6-18-18-4	25,000	0.02
4	1	ST1, ST2, T, V	CL, UTS, TF, RA, CT1	MLP	4-18-18-6	25,000	0.02
5	1	ST1, ST2, T, V	CL, UTS, TF, RA, CT2	MLP	4-18-18-5	25,000	0.02

[*] SOM was trained for 8 classes, 6 points in each class, and a standard deviation of 0.05

Table 2: NN Performance Study of Material Corrosion- Raw Data

Error of Individual Networks				
	Parameters			
NN Model	Crack Length	UTS	Time of failure	Reduction of area
Model 1	0.4236	**0.1460**	0.2361	4.0897
Model 2	0.5963	0.2343	0.1643	5.9212
Model 3	0.4610	0.2335	0.2335	4.5191
Model 4	**0.3321**	0.1634	**0.1146**	3.9106
Model 5	0.4524	0.1481	0.2775	**3.3689**
Ensemble Error				
$\overline{E}$	0.4289	0.1861	0.2167	4.2812
$\overline{A}$	0.2082	0.0608	0.1349	1.6582
E	**0.2496**	**0.1295**	**0.0865**	**2.7733**
min(E)	**0.2305**	**0.1259**	**0.0856**	**2.7456**
Models	1, 3, 4	1, 4, 5	1, 2, 3, 4	1, 2, 4, 5
max(E)	**0.3968**	**0.1646**	**0.1627**	**3.8604**
Models	2, 3	2, 3	3, 5	2, 3

Table 3: NN Performance Study of Material Corrosion- Cleaned Data for RA

Error of Individual Networks				
NN Model	Parameters			
	Crack Length	UTS	Time of failure	Reduction of area
Model 1	0.2529	**0.1204**	**0.0427**	**1.2165**
Model 2	0.8039	0.2237	0.1269	2.8658
Model 3	0.2458	0.1603	0.0804	1.9874
Model 4	**0.1934**	0.2578	0.0746	1.7532
Model 5	0.2107	0.1888	0.0965	1.2779
Ensemble Error				
$\overline{E}$	0.3189	0.1775	0.0865	1.8121
$\overline{A}$	0.1691	0.0603	0.0388	0.9860
E	**0.1659**	**0.1255**	**0.0528**	**0.8612**
min(E)	**0.1317**	**0.1149**	**0.0476**	**0.8612**
Models	1, 3, 4, 5	1, 3	1, 4	1, 2, 3, 4, 5
max(E)	**0.3498**	**0.1867**	**0.0808**	**1.4394**
Models	2, 3	2, 4	2, 5	2, 4

Table 4: Error rates of SG(k-NN) and their ensemble

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**
min(E)	5.0975	2.9443	2.9916
Models	4, 5	4, 5	5, 6
max(E)	6.0197	3.5623	3.5881
Models	1, 2	1, 2	1, 2