Machine learning of material behavior knowledge from empirical data

Yoram Reich & Nahum Travitzky
Department of Solid Mechanics, Materials, and Structures,
Faculty of Engineering, Tel Aviv University, Ramat Aviv 69978, Israel
Phone: +972-3-640-7385, Fax: Phone: +972-3-640-7617
email: yoram@eng.tau.ac.il

March 12, 1996

Abbreviated Head: Learning material behavior knowledge

Key Words: symbolic machine learning, information modeling, modeling material behavior, corrosion

Materials & Design (in press)

Abstract

Symbolic machine learning techniques can extract flexible and comprehensible knowledge from empirical data of material behavior. The diversity of symbolic machine learning techniques offers potential to match the requirements of many tasks when models of material behavior need to be created from data. We develop a series of steps for generating material behavior from empirical data and exemplify some of them on several small datasets. We discuss some of the issues that govern knowledge extraction and as a by-product, demonstrate that symbolic learning techniques are functionally superior to sub-symbolic learning for the task of comprehensible knowledge extraction.

1 Introduction

Much of our knowledge about materials is expressed in phenomenological models created by interpreting empirical data. Such models that correlate between, for example, metal properties and their corrosion rate, can be used for various purposes such as *predicting* the loss of material over time or *guiding* the future selection of materials. We favor models that we can comprehend, that can be integrated with knowledge derived from other empirical data or theories, and that can be used for multiple purposes.

In employing traditional statistical techniques, the process of finding correlations varies from single regression between two sequences of data to fitting complex multivariate surfaces to multidimensional data. The use of statistical techniques requires intimate knowledge of the techniques in order to use them and interpret their results correctly.

This paper presents another set of techniques, called *machine learning* (ML), with their own advantages and difficulties. They also require knowledge for their effective use, but are less complicated to employ and their results are more comprehensible to users. If used intelligibly, ML techniques can deal well with a mix of quantitative, qualitative, missing, or noisy data so common in engineering.

ML techniques are algorithms developed in artificial intelligence that accept *input* information in several forms and transform it into another *form*, referred to as "*knowledge*;" this knowledge captures *important properties* of the input and can be used to explain the input information or be used for problem solving. To illustrate, the input information may be data on material properties, environmental conditions, and behavior of these materials and the form of knowledge may be rules such as:

IF		material is Stainless Steel 304
	and	temperature $< 50.0°$ C
THEN		Stress Corrosion Cracking will not occur

We have to remember that this rule applies only under the same fixed conditions of the experiment.

A subset of ML techniques called *inductive machine learning* is the one relevant to our purpose; it can be subdivided into: *supervised* and *unsupervised concept learning*. Supervised concept learning extracts knowledge for classifying new objects into a set of pre-defined classes. For most ML techniques, examples are represented by a list of property-value pairs and are classified into a set of classes designated by one property. The task is to predict this property based on the remaining properties. Unsupervised concept learning can create clusters or a classification of objects from a given set. This classification can then be used by a supervised concept learning technique to generate classification knowledge.

There are several important aspects governing the successful use of ML techniques in engineering [1] in general, and for modeling engineering information [2] in particular. Some of them are discussed in this paper. Since we desire knowledge structures that are comprehensible, sub-symbolic learning methods such as neural networks are inappropriate. While they can be used to create operational models for prediction, these models remain as "black boxes". In contrast, many symbolic ML techniques can produce comprehensible knowledge structures, the common forms being rules or decision trees. We demonstrate this using two programs: CN2 [3] and OC1 [4].

CN2 is a supervised concept learning program that creates classification rules from training examples represented by a list of property-value pairs. CN2 creates rules by searching for a conjunction of property tests that *covers* a subset of the training examples. Such a conjunct forms the condition part of a classification rule and the classification of the rule is the most common class in the training examples that satisfy the condition. In its search for rule conditions, CN2 specializes conjuncts until one that covers a large number of examples of a single class and few of other classes is located by passing a predictive and reliability evaluations that are governed by operational parameters that the user controls. Once a rule is created, the training examples it covers are removed from further consideration and another rule is sought. This process continues until the training set is exhausted. The product of this process is an ordered set of rules; that is, in classification, the rule set is used in the order in which is was created. CN2 can also create an unordered rule set in which all rules are matched in classification. In this case, conflicting classification can arise from different rules. This is resolved by preferring the rule that covers the largest number of positive training examples. Figure 1 lists rules created by CN2 and Figure 2 provides their graphical interpretation. We will explain these figures in Section 3.1.

OC1 is a supervised concept learning program that creates multivariate decision trees where data is described by numeric properties. Instead of tests on single attributes, as in most common decision tree induction systems, the tests in the decision trees that OC1 creates are based on linear combinations of one or more properties. Thus, these trees partition the space of objects with both oblique and axis-parallel hyperplanes. An example of such partition is shown in Figure 4, Section 3.1.

In the remainder of this paper we discuss how ML techniques can be used to extract knowledge or phenomenological models (Section 2), demonstrate their use for building models of material corrosion (Section 3), and discuss these results.

2 Building empirical models by ML

Suppose we are interested in solving the task of predicting the performance of some material under some conditions and we collect empirical data on various specimens. There are several steps we need to follow in order to build models from this empirical data.

First, we have to *formulate the learning problem in the particular task context.* The task context and what needs to be learned may or may not be fully understood *a priori.* We may conduct some experiments to conclude that we did not collect all the necessary data or that our models are only partial. In real problems, the task understanding evolves as the problem is studied. To illustrate, the task context may be the prediction whether stress corrosion cracking will occur under certain conditions which is a classification task, and the learning problem may be the extraction of classification rules from previous data on the behavior of the material in similar conditions.

Second, we have to *prepare input for the ML programs.* The integration of multiple sources of information, such as multiple experiments, is a critical and difficult task. It has significant implication for the success of the learning process. The decisions of how to adapt empirical data to fit the constraints on the input of ML programs is also critical. Once cases are collected from various sources, *a schema for representing them must be devised.* A list of property-value pairs is the traditional representation for most learning programs. It is supposed to capture the properties used to describe cases, their values, and their relative importance.

Third, a *learning program is selected.* In principle, this selection is based on the properties of the learning problem created after simplifying the task context. It is necessary to assess the relevance of this selection by testing the learning program in the context of the original task. Thus, if a program is selected to learn classification knowledge, this knowledge must be tested against new data to verify its correctness. In most cases, however, programs are selected because they are readily available and not because they are actually the best for the task. In many cases, the representation "supposedly natural" to the domain, is selected to fit the learning program available.

Fourth, *operational parameters for the learning program are selected.* This selection has significant impact on the results [5]. In the course of modeling, many parameters should be experimented with. Most ML systems have several operational parameters that govern their behavior. These may include threshold values for handling noise in data or types of criteria for selecting how to search in the space of rules or trees that the programs generate. Different parameter values give rise to different models. In order to select the operational parameters that lead to the best models, we test the program with these parameters in the best performance test for ML programs: the *leave-one-out* test [6].

Fifth, *the results of models are analyzed.* Most often, the quality of learning is evaluated based on the reduction of errors in arriving at correct solutions. For the purpose of building comprehensible models, other evaluations such as expert ratings are as important [7].

3 Extracting empirical models of corrosion behavior

In this section we demonstrate the effectiveness of symbolic ML techniques for constructing empirical models of corrosion behavior of metals. We use several previously published sets of data for comparison purposes and denote them by: Data1 (Table 1 from [8]), Data2 (Table 2 from [8]), and Data3 (Table 2, p. 175, from [9]). These tables are reproduced as Table 1, 2, and 3 in the Appendix.

The learning task is to build descriptive as well as performance models for describing and predicting when stress corrosion cracking occurs. Given this focus, each data can be treated as an example for training ML programs where the fate of the specimen (whether it cracked or not) determines the class of the example for the ML program. Other potential outcomes such as loss of material (as in Data3), are left out from analysis. Their treatment requires different formulation of the learning problem. (Note that for statistical analysis, multiple dependent variables also require different treatment than when only one dependent variable exits.)

Since we use existing data sets, the second issue in building models is side stepped. Also, we use two representative symbolic ML programs in order to explain some of their characteristics, thus avoiding an in-depth treatment of the third issue.

3.1 Data1

This data relates the stress corrosion cracking (SCC) of a 304 stainless steel to the temperature and $\log CL^{-}$ of the solution (Table 1 from [8]). Smets and Bogaerts suggested the use of a neural network for extracting corrosion knowledge from this data. They trained a network and then used it to predict the SCC values of unseen examples. They interpret the value of the output node of the network as a *reliability* measure. Based on the results, they plotted "equi-reliable" lines and used the plot to interpret the knowledge implicitly embedded in the network.

Clearly, such interpretation is possible to perform on simple problems (i.e., where the data can be plotted in 2-D). It can be much harder even with two parameters and almost impossible when the problem involves three or more parameters. Also, the interpretation of output node value as a measure of performance of the knowledge for plotting the reliability curves is *ad hoc*. We have already stated that for the purpose of extracting comprehensible knowledge sub-symbolic approaches are inappropriate. (Notwithstanding recent studies for extracting rules from neural networks weight data.) Smets and Bogaerts pointed correctly to the critical issue of preparing data when they modified the Cl^- concentration data to its logarithm. This corresponds to our aforementioned second issue.

We apply both CN2 and OC1 to Data1. Figure 1 lists a set of eight rules created by CN2 from Data1. Each rule has a premise and a conclusion part, except for the default rule which is used when no other rule applies. For each rule, the number in square brackets specifies the number of cases from each class (i.e., `[SCC = no SCC = yes]`) the rule covers. A rule that covers more examples from the class it predicts is "stronger" than another that covers less. The figure provides two measures of model "goodness": (1) model *predictive accuracy* as estimated by a leave-one-out experiment, and (2) model *descriptive accuracy* measured by testing the model on the same examples used to create it. These tests are also referred to as *performance* and *coverage* measures [10], or *re-sampling* and *re-substitution* in statistics terminology. The first test is an excellent indicator of a classifier ability to classify new objects. The second test describes how well does the knowledge structure capture the training examples, and is an optimistic estimation of the classifier performance.

Figure 2(a) shows a graphical presentation of the rules predicting `SCC`, Figure 2(b) shows those predicting `No SCC`, and Figure 2(c) shows areas of overlap between the rules that are resolved by selecting the "stronger" rule; and areas not covered by any rule, thus leaving the default rule (1.8) to apply. The first rule captures Smets and Bogaerts' observation that when the temperature is below 50° no corrosion occurs and rules 1.2 and 1.4 capture their observation that at low concentration of Cl^-, no corrosion occurs even at high temperatures.

We can observe some difficulties in the rules. For example, if we compare rules 1.2 and 1.7

```
1.1   IF    temperature < 50.00
      THEN  SCC = no  [6 0]
1.2   IF    270.00 < temperature < 360.00
        AND log_CL < 0.44
      THEN  SCC = no  [4 0]
1.3   IF    temperature < 197.50
        AND log_CL < 0.83
      THEN  SCC = no  [3 0]
1.4   IF    log_CL < -0.50
      THEN  SCC = no  [3 0]
1.5   IF    temperature > 50.00
        AND log_CL > 0.39
      THEN  SCC = yes  [0 16]
1.6   IF    temperature > 360.00
      THEN  SCC = yes  [0 1]
1.7   IF    227.50 < temperature < 270.00
        AND log_CL > -0.26
      THEN  SCC = yes  [0 8]
1.8   IF    none of the above rules apply
      THEN  SCC = yes  [13 18]
```

Model accuracy: Predictive 80.00%, Descriptive 100%

Figure 1: Rules extracted from Data1 by CN2.

we are surprised. In a case where log_CL < 0.30, rule 1.2 tells us that in high temperatures no SCC occurs, whereas rule 1.7 suggests that in lower temperatures SCC will occur. The reason for this surprise can be observed from the plotted data (Figure 2; but we can do it only in 2-D datasets). One SCC data point (#17 from Table 1, [8]), (0.0, 260.0), is suspect of being a mistake. If we remove it, we get a much more accurate model and concise rules (Figure 3).

Figure 4(a) and (b) depict graphical representation of decision trees created by OC1 from Data1 and after removing the case suspect of being an error from Data1, respectively. Figure 5 shows the structure of the decision tree depicted in Figure 4(b). The equation of the line at the Root (i.e., $0.387Cl^- + 0.00286T = 1$, Figure 5) corresponds to the line marked by Root in Figure 4(b). This line divides the space into two regions that are represented by the left an right branches. The subsequent nodes, $T = 50$ and $Cl^- = 1.35$ correspond to the lines marked by l and r in Figure 4(b), respectively. Each such line further subdivides the space into two regions. This continues until some stopping criteria is reached. IN the simplest case, the subdivision stops when a region contains data from one class only: SCC or No-SCC. The number of data points in each such region is given at the leaves of the tree. In

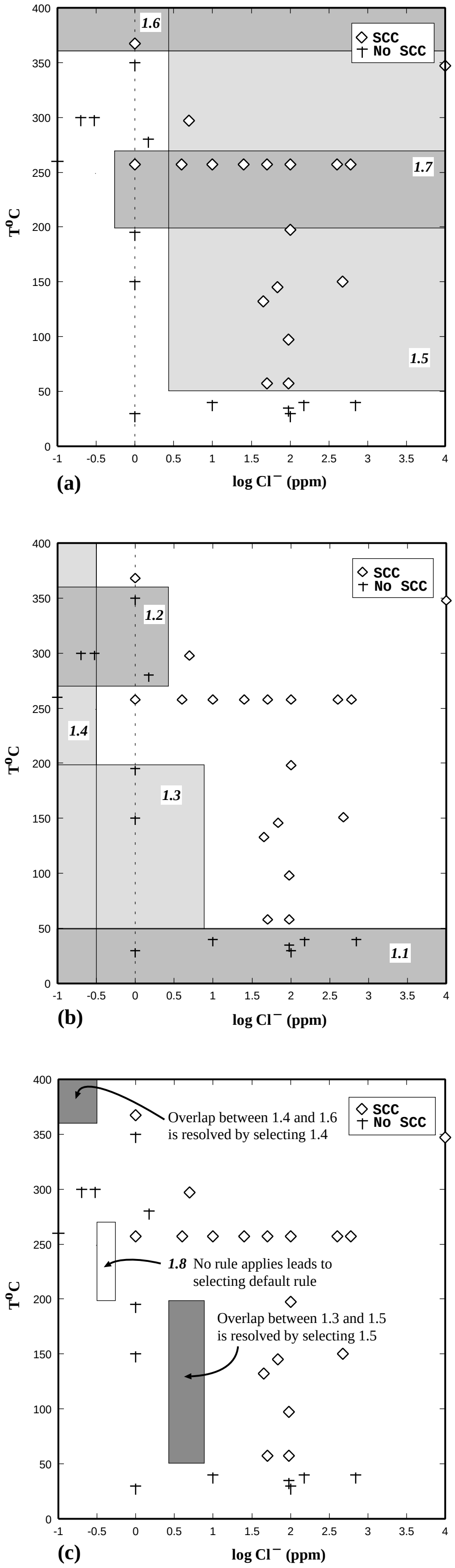

Figure 2: Graphical presentation of SCC and No-SCC rules created by CN2 on Data1.

```
1.9   IF    temperature < 360.00
        AND log_CL < 0.39
      THEN  SCC = no   [8 0]
1.10  IF    temperature < 50.00
      THEN  SCC = no   [6 0]
1.11  IF    temperature > 50.00
        AND log_CL > 0.39
      THEN  SCC = yes  [0 16]
1.12  IF    temperature > 360.00
      THEN  SCC = yes  [0 1]
1.13  IF    none of the above rules apply
      THEN  SCC = yes  [13 17]
```

Model Accuracy: Predictive 93.10%, Descriptive 100%

Figure 3: Rules extracted from Data1, less suspect example, by CN2.

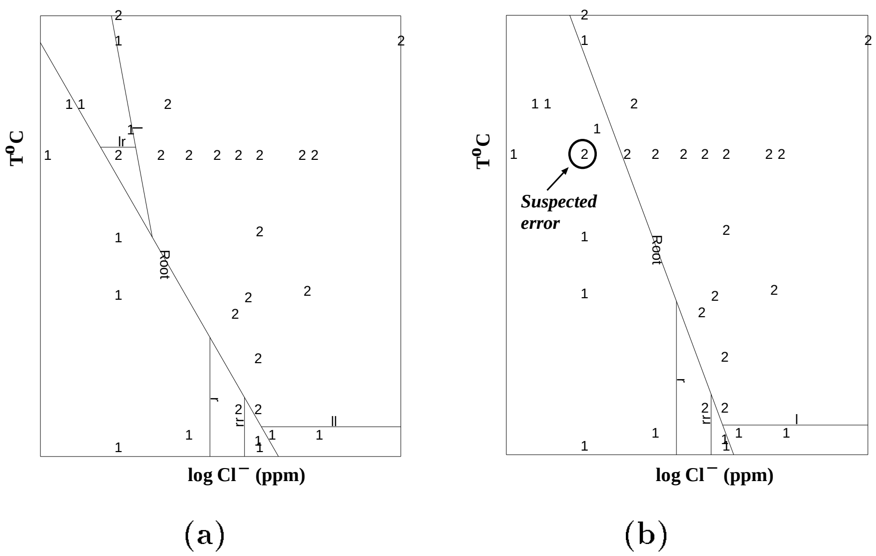

Figure 4: Graphical presentation of decision trees created by OC1 from Data1 and Data1-modified.

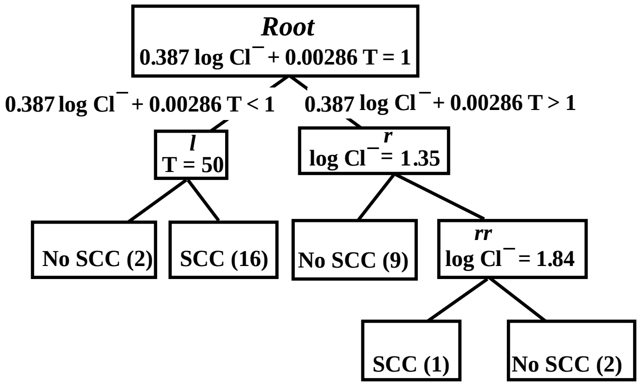

Figure 5: Decision tree created by OC1 from Data1-modified.

our case, the tree has 100% accuracy since all the classes are partitioned exclusively between the leaves.

OC1 tree is less comprehensible than CN2 rules. However, OC1 partitions the domain with hyperplanes instead of axis-parallel planes like CN2 and most other ML programs. This additional degree of freedom enables OC1 to better model relationships between data; although in our example, it is not more concise than CN2 rules. The reason for the use of OC1 here is to show how a hyperplane can be drawn automatically from the knowledge extracted without resorting to prediction tests and result interpretation. In this case, the graphic presentation is more comprehensible than the knowledge structure itself.

3.2 Data2

This data relates the stress corrosion cracking (SCC) of a 304 stainless steel to the $\log CL^-$ and $\log O_2$ of the solution, at high temperatures [8]. Figure 6 lists the rules created by CN2 from Data2. As before, they can be easily interpreted and understood. Figure 7 shows the graphical representation of a decision tree created by OC1 from Data2. We see three data points that we might want to verify their correctness or explain their location.

There are several interesting observations from the rules extracted from Data1 and Data2. If temperature > 50.00, we can consider rules 1.5 and 2.2. The first tells us that for log_CL > 0.39, SCC occurs, while the second contradicts it by saying that for some log_O_2 conditions, (log_O_2 < -0.59), even if log_CL > 0.85, SCC does not occur. Thus, for example, if the temperature = 275°C, log_CL = 0.9, and log_O_2 = -0.60, we have a conflict between the knowledge acquired from the two different sources. The reason for this apparent contradiction

```
2.1   IF    log_O2 < 0.89
        AND log_CL < 0.39
      THEN  SCC = no  [10 0]
2.2   IF    log_O2 < -0.59
        AND log_CL > 0.85
      THEN  SCC = no  [6 0]
2.3   IF    -0.31 < log_O2 < 0.40
        AND log_CL < 1.98
      THEN  SCC = no  [5 0]
2.4   IF    log_CL < -0.50
      THEN  SCC = no  [5 0]
2.5   IF    log_O2 > 0.54
        AND log_CL > -0.50
      THEN  SCC = yes  [0 17]
2.6   IF    log_O2 > -0.56
        AND log_CL > 2.28
      THEN  SCC = yes  [0 6]
2.7   IF    0.44 < log_CL < 0.85
      THEN  SCC = yes  [0 4]
2.8   IF    none of the above rules apply
      THEN  SCC = yes  [19 20]
```

Model accuracy: Predictive 84.21%, Descriptive 100%

Figure 6: Rules extracted from Data2 by CN2.

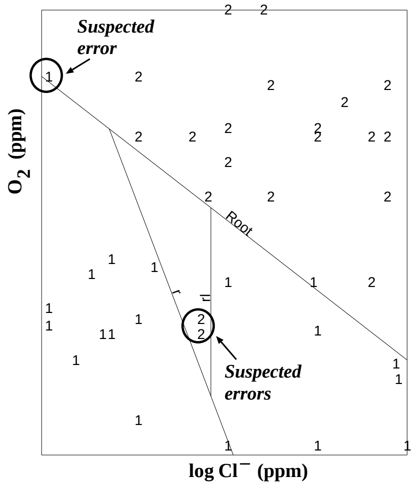

Figure 7: Graphical presentation of decision trees created by OC1 from Data2.

is that in Data1, a critical determinant of corrosion, the log_O_2, was left out from the data and *overgeneral* knowledge was acquired. Also, rules 1.7 and 2.1 contradict each other on a case where: temperature = 260°C, log_CL = -0.2, and log_O_2 = 0.85; again due to omitting the log_O_2 information in Data1.

Some techniques such as CN2, can deal with missing data. We can combine Data1 and Data2 together in several ways and use CN2 to extract knowledge from the combined database, a task that Smets and Bogaerts pose as future work in their paper. This exercise briefly touches upon the data preparation issue.

3.3 Data1 and Data2 combined

This section illustrates two basic methods for combining the data. If we combine the two datasets and leave the unspecified data (i.e., log_O_2 in Data1, and temperature in Data2) as missing, we can get the rules in Figure 8.

Rule 3.4 captures the contribution from Rule 1.5 and uses the data from Data2 to constrain it and consequently, resolves the contradiction. Now, the data point temperature = 275°C, log_CL = 0.9, and log_O_2 = -0.60, does not match any rule and would have to be predicted by the default rule 3.12 as “SCC occurs”. Note that if we remove the suspected case in Data1 as before, the model predictive accuracy will improve to 91.18% and the descriptive accuracy will be the same.

The second method for combining the data involves specifying the temperature values in Data2 as the midpoint in that experiment (275°C), we get better rules due to the additional, even though approximate information (Figure 9); the rules are more specific in that each covers less data points than before (e.g., Rule 3.18 versus 3.4 and Rule 3.20 versus 3.6). Rule 3.18 captures the data from rule 1.5, and is more general now than the rule from the data with the missing values. Nevertheless, the whole model has lower predictive accuracy even though a higher descriptive accuracy. Rule 3.13 predicts that in the case specified before, SCC will not occur. This seems to be a correct prediction, although, the true fate of a metal in these conditions can only be verified in real experiments.

```
3.1   IF     log_CL < -0.15
      THEN   SCC = no  [11 0]
3.2   IF     temperature < 50.00
        AND  log_O2 < -0.59
        AND  log_CL > 0.85
      THEN   SCC = no  [5.50 0]
3.3   IF     log_O2 < 0.84
        AND  log_CL < 0.39
      THEN   SCC = no  [14 1]
3.4   IF     temperature > 50.00
        AND  log_O2 > 0.00
        AND  log_CL > 0.39
      THEN   SCC = yes  [0 15.50]
3.5   IF     temperature > 230.00
        AND  log_O2 > 0.00
        AND  log_CL > 0.39
      THEN   SCC = yes  [0 12]
3.6   IF     0.39 < log_CL < 0.89
      THEN   SCC = yes  [0 6]
3.7   IF     2.24 < log_CL < 2.81
      THEN   SCC = yes  [0 9]
3.8   IF     1.20 < log_CL < 1.90
      THEN   SCC = yes  [0 8]
3.9   IF     temperature > 360.00
        AND  log_O2 > 1.45
        AND  log_CL > -0.50
      THEN   SCC = yes  [0 7]
3.10  IF     none of the above rules apply
      THEN   SCC = yes  [32 38]
```

Model Accuracy: Predictive 88.41%, Descriptive 94.3%

Figure 8: Rules extracted from Data1 and Data2 combined.

```
3.13  IF    267.50 < temperature < 360.00
        AND log_O2 < 0.89
        AND log_CL < 0.39
      THEN  SCC = no  [12 0]
3.14  IF    temperature < 50.00
      THEN  SCC = no  [6 0]
3.15  IF    temperature > 267.50
        AND log_O2 < -0.59
        AND 0.85 < log_CL < 3.50
      THEN  SCC = no  [6 0]
3.16  IF    log_CL < -0.26
      THEN  SCC = no  [11 0]
3.17  IF    267.50 < temperature < 290.00
        AND -0.31 < log_O2 < 0.40
        AND log_CL < 2.28
      THEN  SCC = no  [5.25 0]
3.18  IF    temperature < 197.50
        AND log_CL < 0.83
      THEN  SCC = no  [3 0]
3.19  IF    temperature > 277.50
        AND log_CL < 0.39
      THEN  SCC = no  [4 1]
3.20  IF    temperature > 50.00
        AND log_O2 > 0.54
        AND log_CL > 0.39
      THEN  SCC = yes  [0 23]
3.21  IF    50.00 < temperature < 267.50
        AND log_CL > 0.30
      THEN  SCC = yes  [0 14]
3.22  IF    0.44 < log_CL < 0.85
      THEN  SCC = yes  [0 6]
3.23  IF    227.50 < temperature < 277.50
        AND log_O2 > 0.89
        AND log_CL > -0.50
      THEN  SCC = yes  [0 21]
3.24  IF    temperature > 157.50
        AND log_O2 > -0.25
        AND log_CL > 2.39
      THEN  SCC = yes  [0 6.50]
3.25  IF    temperature > 360.00
      THEN  SCC = yes  [0 1]
3.26  IF    none of the above rules apply
      THEN  SCC = yes  [32 38]
```

Model Accuracy: Predictive 82.61%, Descriptive 98.6%

Figure 9: Rules extracted from Data1 and Data2 combined and midpoint temperature by CN2.

3.4 DATA3

Figure 10 shows the rules extracted by CN2 from Data3 [9], without the loss of thickness data. If we wanted to incorporate the loss of thickness in the analysis we would have to formulate the learning problem differently. We would have started with clustering the data (by unsupervised learning) followed by the application of a supervised concept learning technique on the data classified by the clustering. The rationale behind such decision lies in the nature of the mapping between the independent and the dependent variables: with the loss of thickness data it is a many-to-many mapping, without it, it is a many-to-one mapping [11].

```
4.1   IF    electrodes = un
      THEN  crack = n  [15 0]
4.2   IF    cacl2 < 1.50
      THEN  crack = n  [10 0]
4.3   IF    st_al < 1.75
      THEN  crack = n  [4 0]
4.4   IF    cacl2 > 1.50
        AND st_al > 1.75
        AND electrodes = c
      THEN  crack = y  [0 13]
4.5   IF    none of the above rules apply
      THEN  crack = n  [22 13]
```

Model Accuracy: Predictive and Descriptive 100%

Figure 10: Rules extracted from Data3 by CN2.

4 Discussion

We presented the results of using two ML programs on few simple datasets. These datasets were used to explain the operation of the programs; however, ML programs can deal with much larger databases. The following observations can be made:

- Symbolic ML techniques can extract comprehensible knowledge, and can point to needs to collect additional data points (potentially, to refute erroneous data (rules 1.2 versus 1.7)).
- Symbolic learning are trained relatively quickly, thus model accuracy can be estimated using established statistical tests such as the leave-one-out experiment. Quick learning

allows for experimentation with the representation of data for improving the results of learning.

- Symbolic ML techniques can model data that has noise, missing property-values, and a mix of qualitative and quantitative properties.
- Symbolic ML techniques can integrate several data sets that otherwise may create conflicting knowledge.
- Symbolic ML techniques can create comprehensible knowledge that can be integrated with other knowledge or be used immediately for prediction.
- Although not demonstrated, Symbolic ML techniques can be integrated with prior knowledge such as: which properties should precede others when creating rules.
- ML techniques can be used as tools to assist the analysis of data, and not to replace in any way the performance and recording of accurate experimental data.

Our tests used existing datasets collected for various purposes. The availability of ML techniques as modeling tools opens new opportunities that must be taken into account when recording data from experiments: Data should be collected with the goal of using it for potentially different analyses than for which they were collected. Therefore, precise experimental conditions including: method, techniques, and equipment should be specified, as well as all other relevant contextual information. Raw data must be kept as much as possible without, for example, performing discretizations of continuous data, or summarizing it. Such data will allow for the use of data from different experiments when using ML techniques.

Once experience with the use of symbolic ML techniques in modeling material behavior is accumulated, specific guidelines for data collection will emerge.

5 Conclusions

We have demonstrated the use of symbolic ML techniques as tools for modeling material behavior data. We described the process of knowledge extraction by ML techniques and illustrated it for building models of the corrosion behavior of metals.

We have illustrated some of the steps in building material behavior models by ML techniques. We have shown that data preparation and technique selection have significant impact on the knowledge extracted. Nevertheless, the use of multiple ML techniques to model the same information provides different perspectives that improves the understanding of the knowledge embedded in the data.

In future studies we plan to expand on the first issue: formulating the learning problem, to include problems of clustering, and multiple-dependent variable handling.

We plan to further develop flexible techniques for modeling material behavior from material databases that will be usable to practitioners. We hope that by demonstrating the potential of these techniques, we can impact and improve the design of material information systems.

6 Acknowledgments

We would like to thank P. Clark and R. Boswell for allowing us access to the CN2 code, and to S. K. Murthy, S. Kasif, and S. Salzberg for the access to the OC1 code. This research was partially supported by The Israeli Ministry of Science and The Arts.

References

[1] Y. Reich, S. Konda, S. N. Levy, I. Monarch, and E. Subrahmanian, "New roles for machine learning in design," *Artificial Intelligence in Engineering*, vol. 8, no. 3, pp. 165–181, 1993.

[2] Y. Reich, "Modeling engineering information with machine learning," *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, vol. 10, no. 2, p. (in press), 1996.

[3] T. Clark and P. Niblett, "The CN2 induction algorithm," *Machine Learning*, vol. 3, no. 4, pp. 261–283, 1989.

[4] S. K. Murthy, S. Kasif, and S. Salzberg, "A system for induction of oblique decision trees," *Journal of Artificial Intelligence Research*, vol. 2, pp. 1–33, 1994.

[5] J. Mingers, "An empirical comparison of selection measures for decision-tree induction," *Machine Learning*, vol. 3, no. 4, pp. 319–342, 1989.

[6] S. M. Weiss and I. Kapouleas, "An empirical comparison of pattern recognition, neural nets, and machine learning classification methods," in *Proceedings of The Eleventh International Joint Conference on Artificial Intelligence, Detroit, MI*, (San Mateo, CA), pp. 781–787, Morgan Kaufmann, 1989.

[7] Y. Reich, "Measuring the value of knowledge," *International Journal of Human-Computer Studies*, vol. 42, no. 1, pp. 3–30, 1995.

[8] H. M. G. Smets and W. F. L. Bogaerts, "Deriving corrosion knowledge from case histories: the neural network approach," *Materials & Design*, vol. 13, no. 2, pp. 149–153, 1992.

[9] P. Klieger and J. F. Lamond, eds., *Significance of Tests and Properties of Concrete and Concrete-Making Materials, STP 169C*. Philadelhia, PA, ASTM, 1994.

[10] Y. Reich, *Building and Improving Design Systems: A Machine Learning Approach*. PhD thesis, Department of Civil Engineering, Carnegie Mellon University, Pittsburgh, PA, 1991. (Available as Technical Report EDRC 02-16-91).

[11] Y. Reich and S. J. Fenves, "The formation and use of abstract concepts in design," in *Concept Formation: Knowledge and Experience in Unsupervised Learning* (D. H. J. Fisher, M. J. Pazzani, and P. Langley, eds.), (Los Altos, CA), pp. 323–353, Morgan Kaufmann, 1991.

Appendix: Tables

Table 1: Table 1 from [8]

Data point	temperature (°C)	$\log CL^-$	SCC
1	30.0	0.0	no
2	30.0	2.0	no
3	35.0	1.9777	no
4	40.0	1.0	no
5	40.0	2.1761	no
6	40.0	2.8451	no
7	60.0	1.6990	yes
8	60.0	1.9777	yes
9	100.0	1.9777	yes
10	135.0	1.6532	yes
11	148.0	1.8388	yes
12	150.0	0.0	no
13	153.0	2.6739	yes
14	195.0	0.0	no
15	200.0	2.0	yes
16	260.0	-1.0	no
17	260.0	0.0	yes
18	260.0	0.6021	yes
19	260.0	1.0	yes
20	260.0	1.3979	yes
21	260.0	1.6990	yes
22	260.0	2.0	yes
23	260.0	2.6021	yes
24	260.0	2.7782	yes
25	280.0	0.1761	no
26	300.0	-0.6990	no
27	300.0	-0.5229	no
28	300.0	0.6990	yes
29	350.0	0.0	no
30	350.0	4.0	yes
31	370.0	0.0	yes

Table 2: Table 2 from [8]

Data point	$\log O_2$	$\log CL^-$	SCC
1	-2.0	1.0	no
2	-2.0	2.0	no
3	-2.0	3.0	no
4	-1.699	0.0	no
5	-1.222	2.903	no
6	-1.046	2.875	no
7	-1.0	-0.699	no
8	-0.699	-0.398	no
9	-0.699	-0.301	no
10	-0.699	0.699	yes
11	-0.658	2.0	no
12	-0.602	-1.0	no
13	-0.523	0.0	no
14	-0.523	0.699	yes
15	-0.398	-1.0	no
16	-0.097	1.0	no
17	-0.097	1.954	no
18	-0.097	2.602	yes
19	0.0	-0.523	no
20	0.079	0.176	no
21	0.176	-0.301	no
22	0.903	0.778	yes
23	0.903	1.477	yes
24	0.903	2.778	yes
25	1.301	1.0	yes
26	1.602	0.0	yes
27	1.602	0.602	yes
28	1.602	2.0	yes
29	1.602	2.602	yes
30	1.602	2.778	yes
31	1.699	1.0	yes
32	1.699	2.0	yes
33	2.0	2.301	yes
34	2.199	1.477	yes
35	2.199	2.778	yes
36	2.301	-1.0	no
37	2.301	0.0	yes
38	3.079	1.0	yes
39	3.079	1.398	yes

Table 3: Table 2, p. 175, from [9]

Data point	Cement	Alkalies as Na_2O, % %	$CaCl_2 2H_2O$, % by weight of cement	Ratio of Steel area to Aluminum area	Electrodes	Loss in surface, thickness, mils	Crack
1	c	0.24	0	28	uncoupled	0.16	no
2	c	0.24	2	14	coupled	0.92	yes
3	c	0.24	2	28	coupled	1.2	yes
4	c	0.24	4	7	coupled	1.6	yes
5	c	0.24	4	14	coupled	2.3	yes
6	c	0.24	4	28	coupled	2.4	yes
7	c	0.24	0	28	uncoupled	0.09	no
8	c	0.24	2	14	uncoupled	0.07	no
9	c	0.24	2	28	uncoupled	0.1	no
10	c	0.24	4	7	uncoupled	0.07	no
11	c	0.24	4	14	uncoupled	0.07	no
12	c	0.24	4	28	uncoupled	0.04	no
13	c	0.24	0	0	?	0.09	no
14	c	0.24	4	0	?	0.04	no
15	d	0.89	0	28	coupled	0.12	no
16	d	0.89	1	3.5	coupled	0.54	no
17	d	0.89	1	7	coupled	0.77	no
18	d	0.89	1	14	coupled	1.0	no
19	d	0.89	1	28	coupled	0.85	no
20	d	0.89	2	3.5	coupled	1.0	yes
21	d	0.89	2	7	coupled	1.4	yes
22	d	0.89	2	14	coupled	1.4	yes
23	d	0.89	2	28	coupled	1.6	yes
24	d	0.89	4	3.5	coupled	1.5	yes
25	d	0.89	4	7	coupled	1.7	yes
26	d	0.89	4	14	coupled	2.2	yes
27	d	0.89	4	28	coupled	3.3	yes
28	d	0.89	0	28	uncoupled	0.33	no
29	d	0.89	2	14	uncoupled	0.06	no
30	d	0.89	2	28	uncoupled	0.06	no
31	d	0.89	4	7	uncoupled	0.07	no
31	d	0.89	4	14	uncoupled	0.09	no
33	d	0.89	4	28	uncoupled	0.08	no
34	d	0.89	0	0	?	0.17	no
35	d	0.89	4	0	?	0.05	no