The Role of Genetic Programming in Describing the Microscopic Structure of Hydrating Plaster

Judith E. Devaney
National Institute of
Standards and Technology
100 Bureau Drive, Stop 8911
Gaithersburg, MD 20899-8911

John G. Hagedorn
National Institute of
Standards and Technology
100 Bureau Drive, Stop 8911
Gaithersburg, MD 20899-8911

Abstract

We apply genetic programming in conjunction with other machine learning methods to obtain concise rules that accurately identify scientifically meaningful components in hydrating plaster over multiple time periods. Genetic programming enables the derivation of understandable rules from otherwise opaque classifications.

Our study was based on three dimensional data obtained through X-ray microtomography at five times in the hydration process. Starting with statistics based on locality and output from an unsupervised classification system (autoclass), we use genetic programming to derive simple rules for identifying three classes. These rules are tested on a separate subset of the plaster datasets that had been labeled with their autoclass predictions. The rules were found to have both high sensitivity and high positive predictive value. Genetic programming in conjunction with other machine learning methods enabled us to go from unlabeled data to simple classification rules in a straightforward manner.

1 INTRODUCTION

Plaster of paris is a widely used material of economic importance. For example, the porcelain industry maintains large numbers of plaster molds whose strength, durability, and ability to absorb water impact the industry's costs [Kingrey et al., 1976]. At NIST, we are using our genetic programming system, GPP (Genetic Programming - Procedural) [Devaney et al., 2001, Hagedorn and Devaney, 2001] to begin to understand the structure of plaster at a microscopic scale. We developed GPP at NIST as a generic genetic programming system to address problems from a variety of scientific areas. In this project, GPP is providing us with insight that we were unable to obtain through the use of other machine-learning techniques.

2 BACKGROUND

Plaster powder is formed by calcining gypsum (calcium sulfate dihydrate, $CaSO_4 \cdot 2H_2O$) to form calcium sulfate hemihydrate ($CaSO_4 \cdot \frac{1}{2}H_2O$). The solid plaster is then formed by adding water to the powder (hydration) and allowing the mixture to set. The equations are [Kingrey et al., 1976]:

Calcination:
$$CaSO_4 \cdot 2H_2O = CaSO_4 \cdot \tfrac{1}{2}H_2O + \tfrac{3}{2}H_2O$$

Hydration:
$$CaSO_4 \cdot \tfrac{1}{2}2H_2O + \tfrac{3}{2}H_2O = CaSO_4 \cdot 2H_2O$$

During hydration, an interlocking network of gypsum crystals forms. See Figure 1 for a scanning electron micrograph (900X) [Clifton, 1973] of precipitated gypsum crystals ($CaSO_4 \cdot 2H_2O$). This crystalline network is the foundation of the strength, durability, and absorptivity of the plaster [Kingrey et al., 1976]. There is much to learn about plaster. Even the form of the kinetic equations (fraction of plaster reacted versus time) is not agreed upon [Hand, 1994][Ridge, 1995][Hand, 1995] and would be an interesting problem for genetic programming also. Understanding the process of setting plaster as well as being able to predict its final properties is of scientific as well as economic interest.

For this study, X-ray microtomography has been used to obtain the high resolution of 0.95μ per voxel in three dimensional images of hydrating plaster. Commercial

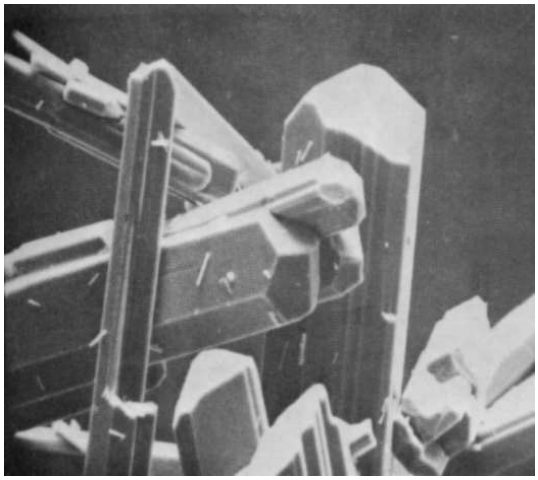

Figure 1: A scanning electron micrograph (900X) [Clifton, 1973] of precipitated gypsum crystals ($CaSO_4 \cdot 2H_2O$).

grade plaster of paris was mixed with a water-to-solids mass ratio of 1.0 and the sample was imaged with X-ray microtomography after 4, 7, 15.5 hours and 6 days. Additionally, a sample of the plaster powder was imaged. This resulted in five images of plaster of size 1024^3. This is gray scale data with each data element varying from 0 to 255 [Bentz et al., 2002][Vis, 2002].

In this project we are looking for simple algorithms to describe and predict components in hydrating plaster. Since classification of plaster does not follow the straighforward methods used for materials such as cement, the algorithms or rules for classifying the elements of the data sets must be found by automatic means. We turned to various machine learning techniques as a means to find these classification rules.

We extracted a 100^3 subset of the data taken at the same place in each of the five time steps. These five data sets were used to develop the rules. A second 100^3 subset was also extracted for validation studies.

The first step in the process was the use of an unsupervised classifier. We use autoclass [Cheeseman et al., 1988] [Cheeseman, 1991] [Stutz and Cheeseman, 1995] [Kanefsky et al., 1994] [Goebel et al., 1989] to do an initial classification of the data. Because materials scientists are interested in three classes within hydrating plaster [Bentz, 2002], we constrain autoclass to seek three classes.

The input to autoclass was a vector of attributes at each image element that was derived from the original data sets. These attributes were selected based on the *Principle of Locality* [Reichenbach, 1932], wherein natural laws are viewed as the consequence of small-scale regularities. Since the structures we are looking for may be as small as a few microns [Vis, 2002], we choose as our scale a 3^3 cube centered on each pixel in the image. Using simple statistics on these small cubes, we create eight attributes for each data element as described in Table 1. These attributes vectors are the input to autoclass.

Autoclass identified three classes that were found to be useful in identifying specific components of the samples. See Figure 2 for a two-dimensional slice of data. On the left is the unclassified intensity data. On the right is the data as classified by autoclass. Images such as this one indicate that autoclass has picked up the basic structure of the data. In the classified data, Class 2 (the white area) is the pore space, Class 1 (the grey area) identifies the crystalline network and the unhydrated plaster, and Class 0 (the black area) represents the boundary region.

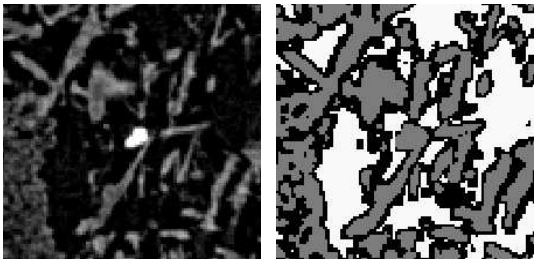

Figure 2: The left image shows a slice of the original plaster data; the classified data is on the right.

But autoclass operates in a "black-box" fashion. The algorithm by which it classifies and predicts elements is opaque to the user. To derive more transparent statements of the autoclass classification schemes, we used a decision tree, C5 [C5, 2002] C5 is the commercial successor to C4.5 [Quinlan, 1993], which has been used extensively for learning.

Table 1: Attributes Used for Classification

Name	Definition
A0	gray level value of pixel itself
A1	neighborhood midrange
A2	nbhd variance about midrange
A3	neighborhood range
A4	neighborhood minimum
A5	neighborhood maximum
A6	neighborhood median
A7	neighborhood mean

In our attempts to derive all of the autoclass classifications using C5, we obtained incomprehensible decision trees with thousands of nodes. These decision trees yielded no insight into the meanings of the classes.

But as we examined the data, we observed that the brightness histograms for the classes suggested that classes 1 and 2 should be easily separable. Using C5, ten fold cross validation on the combined class 1 and 2 showed that this was the case in four of the five datasets. Single node decision stumps with less than five misclassifications over hundreds of thousands of cases were found for the powder, 4 hour, and 15.5 hour data sets. All of these branched on attribute A1. The fourth simple classification was for the 7 hour dataset. This also yielded a single node decision stump; however, this branched on A7. The six day dataset did not yield a simple decision tree for the combined class 1 and 2. For uniformity in the final rules across the hydration times, the 7 hour and six day cases were rerun requiring C5 to use only A1 to get a single best split on this attribute.

The attribute A1 is the local midrange. The midrange is a robust estimator of the center of a short tailed distribution [Crow and Siddiqui, 1967]. Since the range is limited for each neighborhood to $0 - 255$, this is the situation here. So all the class 1/2 discriminator rules are now of the form: *if the center of the local distribution is* $\leq x$ *....*

At this point in the process, we had obtained a good, but opaque, classification scheme from autoclass, and a concise rule for separating class 1 from class 2 from C5. But we had no understandable rules for separating class 0 from class 1 or class 0 from class 2. The decision tree approach had failed to produce useful results for these decisions. It was at this point that we turned to genetic programming.

3 DERIVING CLASSIFICATION RULES WITH GENETIC PROGRAMMING

The class 1/2 decision algorithm generated by C5 for each case separates elements into two set: one set contains class 1 elements and class 0 elements; the other set contains class 2 and class 0 elements. Genetic programming is then used to derive second-tier decision algorithms to separate class 0 from class 1 and class 0 from class 2. We use GPP, a genetic programming system that we have developed here at NIST.

The goal was to derive simple and understandable formulae that closely match the original classifications provided by autoclass.

The method for using GPP in this problem followed the following steps for each desired classification:

- Prepare training data sets from the classified data sets.
- Select parameters, such as the operator set, for the GPP runs.
- Construct a fitness function to measure algorithm effectiveness.
- Execute a set of GPP runs.
- Select the run with results that most closely match the original classification.
- Simplify the GPP-produced algorithm to a succinct form.

3.1 GPP TRAINING DATA SET PREPARATION

The input data to GPP is of the same form as that provided to autoclass except that for each element it includes the classification determined by autoclass. That is, it consists of a set of elements from the plaster volume and for each element it provides the eight attributes described above plus the class to which that element was assigned by autoclass.

As mentioned above, the decision algorithms to be derived by GPP are be used as a second-tier decision after the application of the class 1/2 decision algorithm derived by C5. The class 1/2 decision algorithm classifies elements from classes 1 and 2 very accurately, but the algorithm also assigns elements from class 0 to either class 1 or 2.

We create training sets for the GPP runs based on the input that the decision algorithms will receive. For example, to create the training data set for the class 0/1 decision algorithm we follow these steps:

- Apply the class 1/2 decision algorithm to class 0.
- Select a subset of class 0 elements classified as class 1 for inclusion in the training set.
- Select a subset of all class 1 elements for inclusion in the training set.

We follow the corresponding procedure for the class 0/2 training sets. For all runs the training sets have 200 elements: 100 elements from each of the two classes being considered.

3.2 GPP RUN PARAMETERS AND FITNESS FUNCTION

A variety of operating parameters must be selected for the GPP runs. All runs were done with the same set of parameters. The parameters were selected based on knowledge of the problem to be solved, prior experience with GPP, and a few preliminary test runs.

The primary parameters that were used for all runs are:

- Operators allowed in the programs: +,−,∗,/,<,negate,mod, conditional assignment,and,or,not
- Population size: 500
- Maximum number of generations: 50
- Rates for genetic operations: (crossover : 10%), (mutation : 60%), (prune : 10%), (repair : 10%), (survival : 5%), (new individual : 5%)

The pruning operation [Hagedorn and Devaney, 2001] was considered particularly important in the project. This operation tends to inhibit program bloat and should lead to smaller solution programs. Smaller programs should result in simpler decision algorithms. Some test runs with and without pruning clearly showed that the the use of the pruning operation resulted in substantially smaller programs. Furthermore, the execution times of the GPP runs were greatly reduced because of the smaller sizes of the evolved programs.

The fitness function is based on the correlation between the algorithm's classifications with the actual classifications. The fact that this is a two-valued problem simplifies the calculation of the correlation. We use the formulation by Matthews [Matthews, 1975], which has been used in the context of genetic programming by Koza [Koza, 1994]. The correlation is given by:

$$\frac{T_p T_n - F_n F_p}{\sqrt{(T_n + F_n)(T_n + F_p)(T_p + F_n)(T_p + F_p)}}$$

where:

- T_p is the number of true positives
- T_n is the number of true negatives
- F_p is the number of false positives
- F_n is the number of false negatives

In this context, a positive determination refers to the classification of an element as class 1 or 2 depending on whether the run is for the class 0/1 decision or the class 0/2 decision. The correlation is evaluated based on the execution of an algorithm on the appropriate training set. This correlation value varies from -1 to 1, where 1 indicates perfect performance on the training set and -1 indicated a perfectly incorrect performance on the training set.

Because these decision algorithms are binary in nature, we can turn a very bad algorithm into a very good algorithm simply by inverting each decision. In terms of the correlation value, this means that we can regard a decision algorithm with a correlation of -0.6 to be just as good as an algorithm with correlation +0.6. So our fitness function is the absolute value of the correlation value given above. Each GPP run seeks to evolve an algorithm that maximizes this fitness value.

3.3 EXECUTION OF THE GPP RUNS AND SELECTION OF THE BEST RUN

For each decision algorithm to be derived, five hundred GPP runs were made. Each of these five hundred runs differed only in the seed to the random number generator. On a single processor a set of five hundred runs would typically complete in approximately 5 hours. Given the resources available to us, we easily ran all 12 sets of runs over a single night.

After completing a set of five hundred runs we then have to select the best run for further consideration. Each of the runs produces an output file that contains the evolved decision algorithm in C++ form (although C++ is not the internal representation of the evolved programs).

We implemented a post-processing procedure to aid us in finding the GPP run that produces the best result. For each of the five hundred runs, the procedure executes these steps:

- Extract the C++ version of the algorithm from the output file.
- Compile the C++.
- Run the program on the appropriate full data sets.
- Write out brief performance statistics for that program.

The performance statistics are then easily scanned to select the best run.

3.4 SIMPLIFICATION OF THE GPP ALGORITHMS

Once the best GPP run has been selected, we would like to express the evolved algorithm in a simple form. GPP produces programs that can be a bit obscure and often include elements that do not contribute to the final decision. For this reason, we seek to simplify the GPP evolved algorithms.

This simplification process is, in part, a manual procedure, but we recognized that it would be advantageous to use symbolic computation software as an aid. There are a variety of symbolic computation systems available and we decided to use Maple [Monagan et al., 2000], which can reduce algebraic expressions to simpler forms.

As mentioned above, each GPP run produces an output file that contains a representation of the evolved program in C++ form. In addition to this C++ representation, GPP also writes out the algorithm in a Maple representation. Maple can operate on this representation of the algorithm and simplify it to a single compact formula.

There are some issues in this scenario because some of our internal program representations cannot be easily expressed in Maple's representation. In these cases, some manual intervention is required to complete the simplification. But this is not overly burdensome and Maple has proven to be invaluable in understanding and simplifying the GPP-evolved programs.

4 RESULTS

The relatively simple decision rules were derived for all five time steps and for each of the required classifications. After deriving the rules, we sought to evaluate their effectiveness relative to the original autoclass classification.

4.1 CLASSIFICATION RULES

Recall that for each time step, C5 derived the top-level rule that separates elements into one group with class 1 and class 0 elements and another group with class 2 and class 0 elements. GPP then derived rules for fully separating the classes.

Note that the derived rules are succinct and the entire classification algorithm for a particular time step is quite compact.

Here are the classification algorithms that were derived by C5 and GPP.

4.1.1 POWDER

if $A1 \leq 42.$
  then *(class is either 2 or 0)*
    if $\lfloor .518494A3 + .019318A6 \rfloor = 2$
      then class = 0
      else class = 2
  else *(class is either 1 or 0)*
    if $(7235./A7) \leq (A7 + A4 + A1)$
      then class = 1
      else class = 0

4.1.2 4 HOUR

if $A1 \leq 27.$
  then *(class is either 2 or 0)*
    if $A6 = 0$
      then if $\lfloor 0.5 + .06145A1 \rfloor = 2$
          then class = 0
          else class = 2
      then if $\lfloor 0.5 + .06145A1 + .003373A3 \rfloor = 2$
          then class = 0
          else class = 2
  else *(class is either 1 or 0)*
    if $(6323/A1) < (A0 + A4 + 0.69213A6 + A7)$
      then class = 1
      else class = 0

4.1.3 7 HOUR

if $A1 \leq 42.5$
  then *(class is either 2 or 0)*
    if $\lfloor .527467A1 + .027467A7 \rfloor = 2$
      then class = 0
      else class = 2
  else *(class is either 1 or 0)*
    if $\lfloor 0.5 + 0.008515A7 \rfloor = 1$
      then class = 1
      else class = 0

4.1.4 15.5 HOUR

if $A1 \leq 30.$
  then *(class is either 2 or 0)*
    if $\lfloor .525849(A6 + A3) \rfloor = 2$
      then class = 0
      else class = 2
  else *(class is either 1 or 0)*
    if $A0 \neq 0$
      then if $A4 < (5321.0/A7) - (a0 + a7)$
          then class = 0
          else class = 1

Table 2: Sensitivity and Positive Predictive Values for the Derived Rules.

Dataset	Sensitivity			Positive-Predictive-Value		
	class 0	class 1	class 2	class 0	class 1	class 2
Powder	0.94	0.88	0.97	0.86	0.97	0.95
4 Hour	0.97	0.96	0.97	0.93	0.98	0.997
7 Hour	0.95	0.95	0.99	0.95	0.95	0.99
15.5 Hour	0.96	0.93	0.98	0.93	0.96	0.996
6 Day	0.97	0.96	0.996	0.99	0.95	0.98

else if $A4 < (5321.0/A7) - a3$
then class = 0
else class = 1

4.1.5 6 DAY

if $A1 \leq 28$.
then class = 2
else *(class is either 1 or 0)*
if $A0 + (A1 - (4643/A7)) > 0$
then class = 0
else class = 1

4.2 EVALUATION OF THE RULES

We use *sensitivity* and *positive predictive value* [Lathrop et al., 1993] as metrics to evaluate our derived rules relative to the original classification. A rule can be optimal with respect to a particular classification in two ways. The rule can be very successful at seeing a class when it is there. This is called its *sensitivity*. And the rule can be very successful at identifying the class in the presence of other classes. This is called its *positive predictive value*. Let T_p be the true positives. Let F_p be the false positives. Let F_n be the false negatives. Then:

$$\begin{aligned} \textit{Sensitivity} &= \frac{T_p}{T_p+F_n} \\ \textit{Positive-Predictive-Value} &= \frac{T_p}{T_p+F_p} \end{aligned}$$

In a confusion matrix, *sensitivity* is accuracy across a row; *positive predictive value* is accuracy down a column.

We test our classification rules with a completely different 100^3 subcube of data from each of the five time steps. To test the rules we first compute the same attribute vectors for the new dataset. Then we use the prediction capability of autoclass to label the vectors using the same classification scheme that autoclass derived from the original data subsets. Finally, we use the above rules to create confusion matrices of the predictions for each of the time periods. We derive the *sensitivity* and *positive predictive values* for class and time period. The derived rules are all highly predictive as shown in Table 2.

5 CONCLUSIONS AND FUTURE WORK

Genetic programming has enabled us to derive clear and concise decision algorithms that accurately predict the class of unseen data for hydrating plaster. Furthermore, genetic programming was successfully used in conjunction with other machine learning techniques and was able to solve problems and provide insight in ways that those other techniques could not.

Our work on plaster has just begun, however. First, we would like to develop a better method for validating the classifications. One approach is to generate simulated plaster data sets for which proper classifications are known, for example using computer model microstructures designed to mimic the Plaster of Paris system [Meille and Garboczi, 2001]. We will also be working with an expert to label manually small subsets of the X-ray tomography data sets. These labeled data will then be used for training and validation. This will likely result in refinement of our rules.

Additionally, we would like to derive rules to separate the unhydrated plaster from the gypsum crystals using GPP. Next we would like to develop equations that accurately predict the class regardless of the time of hydration, i.e. that work over the whole hydration period. We will need additional data to include variations with respect to the parameters that can influence the setting process and the resultant properties of plaster. Finally, we would like to predict physical characteristics of classes with equations, instead of predicting classes. We expect genetic programming to be an essential tool throughout these efforts.

Acknowledgments

We would like to thank Dale Bentz for his support and advice. We would also like to acknowledge valuable input and support from Barbara Cuthill and John Hewes, and financial support from the NIST Advanced Technology program.

Disclaimer

Certain commercial products may be identified in this paper in order to adequately describe the subject matter of this work. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the identified products are necessarily the best available for the purpose.

References

[C5, 2002] (2002). C5. [online] `http://www.rulequest.com`.

[Vis, 2002] (2002). The visible cement dataset. [online] `http://visiblecement.nist.gov`.

[Bentz, 2002] Bentz, D. P. (2002). Personal communication.

[Bentz et al., 2002] Bentz, D. P., Mizell, S., Devaney, S. G. S. J. E., George, W. L., Ketcham, P. M., Graham, J., Porterfield, J., Quenard, D., Vallee, F., Sallee, H., Boller, E., and Baruchel, J. (2002). The Visible Cement Dataset. *J. Res. Natl. Inst. Stand. Technol.*, 107(2):137–148.

[Cheeseman, 1991] Cheeseman, P. (1991). On finding the most probable model. In Shrager, J. and Langley, P., editors, *Computational Models of Discovery and Theory Formation*, pages 73–96. Morgan Kaufman, San Francisco, CA.

[Cheeseman et al., 1988] Cheeseman, R., Kelley, J., Self, M., Taylor, W., and Freeman, D. (1988). Autoclass: A bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 65–74, San Francisco, CA. Morgan Kaufman.

[Clifton, 1973] Clifton, J. R. (1973). Some aspects of the setting and hardening of gypsum plaster. Technical Note 755, NBS.

[Crow and Siddiqui, 1967] Crow, E. L. and Siddiqui, M. N. (1967). Robust estimation of location. *Journal of the American Statistical Associatiuon*, 63:363–389.

[Devaney et al., 2001] Devaney, J., Hagedorn, J., Nicolas, O., Garg, G., Samson, A., and Michel, M. (2001). A genetic programming ecosystem. In *Proceedings 15th International Parallel and Distributed Processing Symposium*, page 131. IEEE Computer Society. `http://math.nist.gov/mcsd/savg/papers`.

[Goebel et al., 1989] Goebel, J., Volk, K., Walker, H., Gerbault, P., Cheeseman, P., Self, M., Stutc, J., and Taylor, W. (1989). A bayesian classifiection of the iras lrs atlas. *Astronomy and Astrophysics*, 222:L5–L8.

[Hagedorn and Devaney, 2001] Hagedorn, J. and Devaney, J. (2001). A genetic programming system with a procedural program representation. In *2001 Genetic and Evolutionary Computation Conference Late Breaking Papers*, pages 152–159. `http://math.nist.gov/mcsd/savg/papers`.

[Hand, 1994] Hand, R. J. (1994). The kinetics of hydration of calcium sulphate hemihydrate: A critical comparison of the models in the literature. *Cement and Concrete Research*, 24(5):885–895.

[Hand, 1995] Hand, R. J. (1995). A reply to a discussion by m. j. ridge of the paper: The kinetics of hydration of calcium sulphate hemihydrate: A critical comparison of the models in the literature. *Cement and Concrete Research*, 25(1):225–226.

[Kanefsky et al., 1994] Kanefsky, B., Stutz, J., Cheeseman, P., Taylor, W., and Clifton, J. R. (1994). An improved automatic classification of a landsat/tm image from kansas (fife). Technical Report FIA-94-01, NASA AMES.

[Kingrey et al., 1976] Kingrey, W. D., Bowen, H. K., and Uhlmann, D. R. (1976). *Introduction to Ceramics*. John Wiley and Sons, New York.

[Koza, 1994] Koza, J. R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press.

[Lathrop et al., 1993] Lathrop, R., Erbdster, T., Smith, R., Winston, P., and Smith, T. (1993). Integrating ai with sequence analysis. In Hunter, L., editor, *Aritifical Intelligence and Molecular Biology*, Cambridge, MA.

[Matthews, 1975] Matthews, B. W. (1975). Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta*, 405:442–451.

[Meille and Garboczi, 2001] Meille, S. and Garboczi, E. J. (2001). Linear elastic properties of 2-d and 3-d modles of porous materials made fromeelongated objects. *Mod. Sim. Mater. Sci*, 9:1–20.

[Monagan et al., 2000] Monagan, M. B., Geddes, K. O., Heal, K. M., Labahn, G., Vorkoetter, S. M., and McCarron, J. (2000). *Maple 6 Programming Guide*. Waterloo Maple Inc., Waterloo, Ontario, Canada.

[Quinlan, 1993] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Klauffann, San Mateo.

[Reichenbach, 1932] Reichenbach, H. (1932). *Atom and Cosmos*. Dover Publications, Inc., Mineola, New York. (First published in 1930 as Atom und Kosmos.).

[Ridge, 1995] Ridge, M. J. (1995). A discussion of the paper: The kinetics of hydration of calcium sulphate hemihydrate: A critical comparison of the models in the literature by r. j. hand. *Cement and Concrete Research*, 25(1):224.

[Stutz and Cheeseman, 1995] Stutz, J. and Cheeseman, P. (1995). Bayesian classification (autoclass): Theory and results. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*. AAAI Press, Menlo Park, CA.