in *Machine Learning: A Multistrat*
Approach, Vol. IV, R.S. Michalski
G. Tecuci, (Eds.), Morgan Kaufma
Publishers, San Francisco, CA, 19

18

MULTISTRATEGY LEARNING FROM ENGINEERING DATA BY INTEGRATING INDUCTIVE GENERALIZATION AND GENETIC ALGORITHMS

Jerzy W. Bala
Kenneth A. De Jong
Peter W. Pachowicz
(George Mason University)

Abstract

The size and complexity of most engineering data and an inherent presence of noise in them pose significant difficulties for traditional concept learning systems. This chapter presents a multistrategy system that addresses these issues. The system integrates two forms of learning—inductive generalization and genetic algorithms—in a closed-loop fashion in order to achieve robust concept learning capabilities for a variety of difficult engineering-oriented classification tasks. The learning process cycles through two phases. In the first phase, initial concept descriptions are acquired by running a noise-tolerant extension of the familiar AQ rule induction system. The resulting concept descriptions are not, however, optimal from a performance point of view because of the strong AQ bias to generate simple, cognitively oriented descriptions. Therefore, in the second phase, the perfor-

mance of the current set of descriptions is improved using a genetic algorithm. The tuned concept descriptions are then fed back to the inductive learning program for further simplification and refinement. The effectiveness of this multistrategy approach is illustrated on a set of difficult texture recognition problems taken from the field of computer vision.

18.1 INTRODUCTION

Engineering problems encompass a wide variety of tasks, including manufacturing automation, design automation, automated process planning, and vision-based automated manufacturing. Each of these tasks represents a practical application area for machine learning technology. At the same time, these engineering domains present significant challenges to traditional concept learning systems for several important reasons.

The principle source of difficulty lies in the size, complexity, and noise inherent in typical engineering data sets collected for purposes of concept learning. Consider, for example, the process of extracting sets of examples of various texture classes from low level pixel data for the purpose of learning texture classification rules. In real-world situations, noise occurs naturally at the input level because of the discretization of continuous image data and changing environmental conditions as a result of the properties of the camera (lens distortion) and of defective sensing elements. These noise sources are generally very difficult to model *a priori* in that their distributions are usually unknown and can change over time, thus making it difficult to develop noise filters.

Typically, from this already noisy pixel data, exemplar data sets are constructed in the form of vectors of real-valued features, resulting in non-systematic errors in the extracted feature values as well as in the class information of the exemplar data sets. This process tends to produce quite complex and irregular distributions of the extracted feature values and usually requires the construction of fairly large data sets in an attempt to "smooth out" the noise. It is, therefore, essential in such domains to develop approaches to concept learning that are able to work with large, noisy data sets (Quinlan, 1986; Chien et al., 1991; Pachowicz and Bala, 1991; Bergadano et al., 1992).

A second distinguishing feature of these engineering domains is that the learned concepts are used in recognition modules that are just one part of a much larger system. This means that the learned concepts must satisfy a number of important criteria above and beyond the traditional measures of cognitive simplicity and accuracy on a test set. We again take an example from image understanding: a texture recognition module uses the learned concepts to make classification decisions. These classification decisions are then used by other modules to draw borders between textured surfaces of objects. Next, surface areas are grouped into

objects (e.g., bushes, roads, sky) and are analyzed at a more global level. To be effective in this context, the learned texture concepts must themselves be able to handle noisy data and must not be the source of additional noise to avoid propagation and amplification of errors to the higher levels.

In many practical applications, a system must also meet additional criteria related to overall recognition effectiveness. Examples of such criteria are recognition stability and full concept recognizability. Recognition stability is typically measured by the deviation of class recognition rates from the average recognition rate and reflects the requirement that there must be similar recognition levels for each class. By requiring high recognition stability of a system, for example, we prefer a situation where class A and class B are both recognizable on the same 80% level and discard a situation where class A is recognizable on 65% and class B on 95% level. A full concept recognizability criterion requires that a system must recognize all classes in the training sets at a high level. Even if a system fails to recognize only one class adequately (e.g., 30%), the criterion of full concept recognizability is not satisfied in spite of the fact that the system recognizes all other classes at a very high level.

In this chapter, a multistrategy system is presented that is designed to address the above issues. The presented system integrates two forms of learning—inductive generalization and genetic algorithms—in a closed-loop fashion in order to achieve robust concept learning capabilities for a variety of difficult engineering-oriented classification tasks. The learning process cycles through two phases. In the first phase, initial concept descriptions are acquired by running a noise-tolerant extension of the familiar AQ rule induction system. The resulting concept descriptions are not, however, optimal from a performance point of view because of the strong AQ bias to generate simple, cognitively oriented descriptions. Therefore, in the second phase, the performance of the current set of descriptions is improved using a genetic algorithm (GA). The tuned concept descriptions are then fed back to the inductive learning program for further simplification and refinement. The effectiveness of the developed multistrategy approach is illustrated on a difficult set of texture recognition problems taken from the field of computer vision.

18.2 INTEGRATING INDUCTIVE GENERALIZATION AND GENETIC ALGORITHMS

The difficulties discussed in the previous section provide the motivation for developing an integrated multistrategy learning system. However, in order for such a system be an improvement over single strategy systems, the incorporated strategies must be both compatible in the sense of being able to share concept descriptions and complementary in the sense that the strengths of one counterbalance some of the weakness of the others. In this section, we discuss the properties of two

learning strategies: the traditional inductive generalization approach as typified by AQ-based rule induction systems (Michalski, 1983) and genetic algorithms (De Jong, 1988). We argue that they are well matched for use in a multistrategy system.

Inductive learning algorithms like AQ provide efficient strategies for acquiring simple, cognitively oriented concept descriptions. Typically, their efficiency is achieved via *a priori* models that guide the search toward building plausible concept descriptions and by incorporating a strong bias for simple descriptions. When the (presumably) unknown concepts are well matched to these assumptions, learning is fast and effective.

However, these same features can be a disadvantage in other contexts. The strong biases that produce efficient search can result in descriptions that are only locally optimal. Running the algorithm with reordered learning data or slightly different settings for search control parameters can produce strikingly different concept descriptions. If the learning data are noisy and/or the concepts to be learned are complex, the speed and effectiveness of the algorithm can be affected negatively by the simplicity bias. Similarly, in situations in which there is little or no *a priori* knowledge to guide the search, the speed and effectiveness of learning can fall off quickly.

Finally, inductive learning techniques generally work better on data sets carefully chosen to provide maximum information and no noise because there is usually a strong bias to produce concept descriptions that are complete and consistent with respect to the training set. The presence of noise tends to produce concept descriptions with lots of disjuncts, many of which cover only a few points. Attempting to be complete and consistent with randomly chosen training data leads to overfitting and to poor predictive performance on unseen examples.

Genetic algorithms, on the other hand, are general purpose adaptive search techniques that come with none of the traditional biases for searching the space of concept descriptions. Rather, use performance-oriented feedback to bias the search process in the direction of better performance. This generally results in a more global search procedure than traditional inductive methods but also means that the search time can be considerably longer. Unless explicit biases toward simplicity and consistency are included in the feedback function or in extensions to the algorithm, the concept descriptions produced by GAs will frequently be more complex than those produced by AQ-based systems and less consistent with the training data.

Because GAs maintain a population of samples, they are much less sensitive to noise than other techniques and have been shown to be effective in the presence of high noise levels without assuming any model of noise distribution prior to the learning (Fitzpatrick and Grefenstette, 1988).

This comparison of inductive learning strategies and genetic algorithms suggests that there is an opportunity to combine the two approaches so that the advantages of one learning method can be used to limit disadvantages of the other learning method. The question still remains of how to integrate them in an effective way to achieve cooperative concept learning.

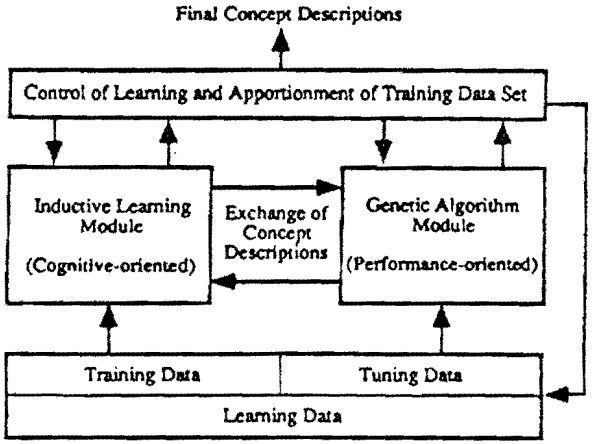

Figure 18.1: An architecture for multistrategy learning from engineering data that integrates inductive learning and genetic algorithms

Our answer to this question is presented in Figure 18.1. The learning process cycles between two phases: an inductive learning phase and a genetic algorithm phase. In the inductive learning phase, cognitively oriented concept descriptions are produced in standard disjunctive normal form (DNF). In the GA phase, the performance of these concepts is improved using a set of tuning data. After the concepts are modified, they are refined again by the AQ algorithm, resulting in somewhat simpler descriptions. In this way, the learning loop is closed and two learning modules are able to exchange concept descriptions and improve them according to different criteria.

By combining inductive learning with genetic algorithms, the learned concept descriptions are no longer required to be complete and consistent with respect to the initial training data, which reduces overfitting problems and leads to better predictive performance. Also, the use of GAs reduces the effects of noise on the learned concept descriptions. The fact that the GA starts its search with plausible AQ-generated concept descriptions results in much shorter search times. Also, the performance-oriented GA search provides the ability to escape from some of the local minima traps resulting from AQ biases.

18.3 THE INDUCTIVE LEARNING MODULE

The inductive learning method used to acquire cognitively oriented concept descriptions is executed in two stages: first, inductive generalization and then con-

cept optimization. The AQ learning program (Michalski, 1973, 1983) is used as the inductive generalization strategy. Concept optimization is then applied in an attempt to decrease the influence of noise on the learned concept descriptions (Pachowicz and Bala, 1991).

18.3.1 AQ Algorithm

The AQ algorithm constructs concept descriptions in a top-down fashion (Michalski, 1973, 1983). The system learns classification rules from preclassified sets of examples (events). Examples are observations from the problem domain, where each example is a vector of attribute-value pairs. The examples of a given class for which the algorithm is learning rules are called the positive examples. Other classes represent negative examples with respect to the selected class. AQ begins with the most general description (the empty, all inclusive conjunctive concept) and specializes the concept by adding one or more particular generalized selectors from the seed (a selected positive example) through the process known as *extend against*. The specialized concept covers fewer negative examples than the more general concept. When the concept is consistent (covers no negative events), it is saved in the current concept set by disjoining it with the previous disjuncts. Next, a new seed is selected from the set of positive examples that are not covered by any of the disjuncts in a concept set. The process continues until all positive examples are covered.

The *extend against* process is used to maximally generalize the values in a condition. To extend the seed value for an attribute against the set of values of the negative set for the attribute, the system determines the most general consistent value subrange for the attribute that includes the seed event's value on that attribute but does not intersect the negative set. In effect, each attribute of the seed is compared with the same attribute of a negative event. In case of linear attributes, *extend against* returns a range of values that include the value for the seed and immediately adjacent values not including the negative event's value. The results of extending the seed against each negative event are then combined by intersecting the values for each of the previous tests for each attribute. The system then selects these combined values to get a disjunct that covers the seed but none of the negative events.

The concept descriptions learned by the AQ algorithm are represented in VL_1, which is a simplified version of the Variable-Valued Logic System VL (Michalski, 1973) and are used to represent attributional concept descriptions. A description of a concept is a VL_1 expression in disjunctive normal form, where each disjunct has the form

[L # R] (1)

where L is called the referee, which is an attribute; R is called the referent, which is a subset of values from the domain of the attribute L; and # is one of the following relational symbols: =, <, >, >=, <=, <>. The following is an example of an AQ disjunct (equality is used as a relational symbol):

[x1=1..3] [x2=1] [x4=0] [x6=1..7] [x8=1] (2)

18.3.2 The AQ-NT Learning Method

The AQ-NT learning method is an extension of the AQ algorithm and is intended to deal with some of the effects of noise on the concepts learned by AQ (Pachowicz and Bala, 1991). The objective is to optimize concept descriptions in two ways: (1) to improve the overall recognition effectiveness of learned descriptions and (2) to improve the generality of concepts fragmented by noisy data. The acquisition of concept descriptions (in the form of a set of decision rules) is performed in the following two phases.

Phase 1: Concept-driven Closed-Loop Filtration of the Training Data. This phase consists of a single loop that attempts to remove items from the data set that appear to contain noise. This process is composed of the following stages:

1. Induce an initial set of concept descriptions from the given dataset using the AQ learning program.
2. Truncate the concept descriptions by removing "least significant" disjuncts, that is, disjuncts that cover only a small portion of the training data.
3. Create a new training dataset that includes only training examples covered by the modified concept descriptions.
4. If the size of the dataset falls below an assumed percentage of the training data (that reflects an assumed error rate in the data), then go to Phase 2. Otherwise, return to step 1.

Phase 2: Acquire Concept Descriptions from the Improved Training Dataset Using the AQ Learning Program. A justification for Phase 1 is that the noise in the data is unlikely to constitute any strong patterns in the data and therefore will require separate disjuncts to account for it. Thus, the examples covered by the "light disjuncts" are likely to represent noise and therefore are removed from the dataset. Experiments with the AQ-NT learning approach (incorporating the NEWGEM program [AQ14] [Mozetic, 1985]) applied to computer vision problems involving texture recognition have shown that it systematically produces classification rules that perform better and also are much simpler in the sense that the number of learned disjuncts is significantly decreased.

18.4 THE GENETIC ALGORITHM MODULE

Genetic algorithms typically maintain a constant-sized population of candidate solutions, known as individuals (Holland, 1975; De Jong, 1988). Search proceeded in an efficient manner by exploiting the information contained in the individuals of the current population. An individual is evaluated and recombined with others on the basis of its overall quality or fitness. This process is iterative, and the expected number of times an individual is selected for recombination is proportional to its fitness relative to the rest of population. By allocating more reproductive occurrences to the above average individuals, the overall effect is an increase in the population's average fitness. New individuals are created using two main genetic recombination operators known as *crossover* and *mutation*. *Crossover* operates by selecting a random location in the genetic string of the parents and concatenating the initial segments of one parent with the final segment of the other parent to create a new child. A second child is simultaneously generated using the remaining segments of the two parents. *Mutation* provides for occasional disturbances in the crossover operation to ensure diversity in the genetic strings over long periods of time and to prevent stagnation in the search process. GAs have been shown to be very effective in situations where no *a priori* information is available about the properties of the space to be searched. In addition, they are quite insensitive to noise because they maintain at all times a population of search space samples.

18.4.2 Optimization of the Concept Descriptions Using Genetic Algorithms

GAs traditionally use fixed-length binary strings to representation individuals. However, if the effective cooperative learning between the AQ-NT module and the GA module must be achieved, it makes much more sense to have the GA work directly on the VL_1 expressions produced by AQ-NT. Fortunately, it is rather straightforward to extend the genetic operators to handle this more complex representation language.

The extended mutation operator is designed to introduce small changes to the condition parts of a selected disjunct of a given description. The conditions to be changed are chosen by randomly generating two pointers: the first one points to one of the disjuncts, and the second one points to a condition in this disjunct. Random changes are introduced to the left-most or right-most (this is also chosen randomly) value of this condition. For example, possible mutations of the condition [x1 = 10..23] might generate any of the following conditions: [x1 = 10..20], [x1 = 10..24], [x1 = 12..23] or [x1 = 8..23], as well as others. Such a mutation process samples the space of possible concept description boundaries to improve the performance criteria.

The second operator, crossover, is performed by splitting concept descriptions into two parts: upper disjuncts and lower disjuncts. These parts are exchanged

between parent descriptions to produce a new child description. An example of crossover applied to two short, four-disjunct parent descriptions is depicted below:

Parent Description 1:

1 [x1=7..8] & [x2=8..19] & [x3=8..13] & [x5=4..54] & [x6=0..3] or
2 [x1=15..54] & [x3=11..14] & [x4=14..17] & [x6=0..9] & x7=0..11] or
——————————crossover position ——————————
3 [x1=9..18] & [x3=16..21] & [x4=9..10] or
4 [x1=10..14] & [x3=13..16] & [x4=14..54] & [x7=4..5] (3)

Parent Description 2:

1 *[x3=18..54] & [x4=16..54] & [x5=0..6] & [x7=5..12] or*
2 *[x1=8..25] & [x3=8..13] & [x4=9..11] & [x5=0..3] or*
——————————crossover position ——————————
3 *[x4=0..22] & [x5=8..9] & [x6=0..7] & [x7=11..48] or*
4 *[x2=5..8] & [x3=7..8] & [x4=8..11] & [x5=0...3]* (4)

One of the Two Resulting Child Descriptions:

1 [x1=7..8] & [x2=8..19] & [x3=8..13] & [x5 =4..54] & [x6=0..3] or
2 [x1=15..54] & [x3=11..14] & [x4=14..17] & [x6=0..9] & [x7=0..11] or
3 *[x4=0..22] & [x5=8..9] & [x6=0..7] & [x7=11..48] or*
4 *[x2=5..8] & [x3=7..8] & [x4=8..11] & [x5=0..3]* (5)

Concept descriptions are evaluated by calculating the degree of match between a given disjunct and an example. Because the degree of match of a given example depends on the degree of match to the "winning" disjunct of that description, the exchange process introduced by crossover enables inheritance of information about strong disjuncts (strongly matching) by the individuals of the next evolved population.

18.4.3 Performance Evaluation

The performance-oriented feedback provided for each concept in the current population is derived from a confusion matrix. The confusion matrix represents information on correct and incorrect classification results. This matrix is obtained by calculating the degree of flexible match between a tuning instance and a given class description. The row entries in the matrix represent a percentage of matched instances from a given class (row index) to all class descriptions (column index). Because some of the testing events can be matched flexibly by more than one class, the sum of the entries in a given row may be larger than 100%. Table 18.1 is an example of the confusion matrix for 12 classes used in our experiments.

Table 18.1: Confusion matrix for 12 classes of texture (average recognition = 74.67%, average mis-classification = 4.83% — from Bala, De Jong, and Pachowicz (1991a). Note: the sum of row entries may be larger than 100% because of multiple matches.

Test Classes	Learned Classes											
	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
C1	45	5	31	0	0	9	0	28	16	0	6	2
C2	5	51	3	12	0	16	1	17	7	0	17	2
C3	31	0	80	0	7	1	0	2	10	0	6	5
C4	0	25	0	72	0	14	0	12	0	0	1	10
C5	0	0	2	0	99	0	0	0	0	0	0	7
C6	8	11	3	25	0	62	0	28	0	0	12	1
C7	0	2	0	0	0	0	98	0	0	0	0	0
C8	19	14	8	2	1	59	0	59	3	0	5	0
C9	16	0	8	0	0	5	0	5	87	0	1	2
C10	0	0	1	0	0	0	0	0	2	95	0	0
C11	*16*	*4*	*32*	*10*	*0*	*4*	*0*	*4*	*2*	*0*	*44*	*7*
C12	0	0	0	4	0	0	1	0	0	0	2	98

The underlined entries represent the percentage of instances of class C11 (set of tuning instances) matched to learned descriptions of all 12 classes. If class C11 is to be optimized by GAs, we have to evaluate the performance of each individual from the population by calculating its performance to the other classes. Thus, we use the ratio of correct recognition rates to incorrect recognition rates as the performance evaluation measure of each individual of the population, i.e., the CC/MC ratio, where CC is an average recognition for correctly recognized classes (the average of the entries on the diagonal of the confusion matrix), and MC is an average mis-classification (the average of the entries outside the diagonal of the confusion matrix). For the above confusion matrix, CC/MC = 74.67/4.83 = 15.45.

The performance evaluation is based on computing a distance measure between an example (tuning or testing) and a rule. A flexible matching technique is applied to calculate such a distance. This distance measure takes any value in the range from 0 (i.e., does not match) to 1 (i.e., matches). The calculation of this distance measure for a single test instance is executed according to the following schema. For a given condition of a rule $[x_a = val_j]$ and an instance where $x_a = val_k$, the normalized distance measure is

$$1 - (| val_j - val_k | / \#levels) \quad (6)$$

where #levels is the total number of attribute values. The confidence level of a rule is computed by multiplying evaluation values of each condition of the rule. The total evaluation of class membership of a given test instance is equal to the confidence level of the best matching rule, i.e., the rule of the highest confidence value. For example, the confidence value c for matching the following complex of a rule

$$[x1 = 0] [x7 = 10] [x8 = 10..20] \quad (7)$$

with a given instance x = < 4, 5, 24, 34, 0, 12, 6, 25 > and the number of attribute values #levels=55 is computed as follows:

$$c_{x1} = 1 - (10 - 4 | / 55) = .928 \quad (8)$$
$$c_{x7} = 1 - (110 - 6 | / 55) = .928 \quad (9)$$
$$c_{x8} = 1 - (120 - 5 | / 55) = .91 \quad (10)$$
$$c_{x2}, c_{x3}, c_{x4}, c_{x5}, c_{x6} = 1 \quad (11)$$
$$c = c_{x1} * c_{x2} * c_{x3} * c_{x4} * c_{x5} * c_{x6} * c_{x7} * c_{x8} = 0.78 \quad (12)$$

Once all rule confidence levels have been computed, the recognition process decides class membership on the basis of the highest confidence level among the matched rules.

18.5 OVERALL SYSTEM ARCHITECTURE

The multistrategy system presented in this chapter is a loosely coupled integration of the two learning components discussed above. The overall architecture of the system is presented graphically in Figure 18.2.

The training data are split randomly into two separate datasets of learning examples. The first dataset serves as the training data for the inductive learning module of the system to generate cognitively oriented concept descriptions. The second dataset is used as tuning data by the genetic algorithm module to optimize rules according to the previously described performance criteria.

The cognitively oriented concept descriptions produced by AQ-NT from the initially provided training data are forwarded directly to the GA module, which then proceeds to optimize the concept descriptions according to the given performance criteria. The concept descriptions forwarded to the GA module are tested on the tuning data in order to acquire their performance evaluation. The "weakest" concept description is then identified and is used as input to the concept evolution system. The concept evolution system initializes the population with variations of this "weakest" concept description and uses a GA to find better performing variations of the description. Each candidate is evaluated jointly with rules describing other classes. The descriptions of the other classes are not simultaneously optimized. The best version found is used to replace the current "weakest" description. The effect of this approach is to improve the performance of the weakest concept description without weakening the others.

The modified set of class descriptions is returned back to the inductive learning module for further simplification. This is accomplished by an interface that performs concept-based data filtration. This filtration step uses the modified concept descriptions produced by the GA module to filter the training data that will be used in the next learning loop. In the filtration step, initial training examples are

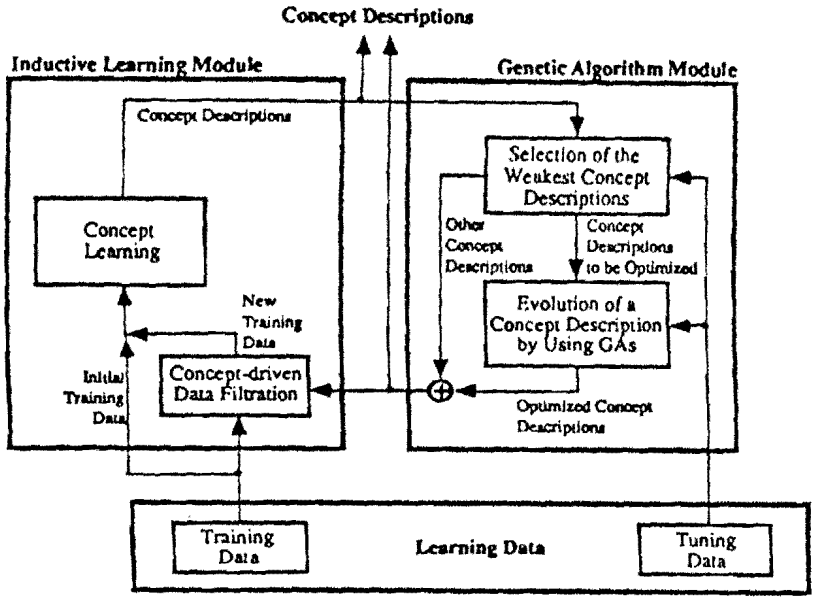

Figure 18.2: Closed-loop cooperation between inductive learning and genetic algorithm modules

matched with the new set of rules. Examples that are covered by the new ruleset (examples that are consistent and complete with rules) form a new training dataset, which is input again to the AQ program. Examples not matched are discarded. In this way, the training dataset used by AQ-NT is modified over time in such a way that performance-oriented constraints in the training data set are preserved, and training examples that have a negative influence on learning performance-optimal rules are rejected.

The learning loop is repeated again but with the filtered set of training data. The expected overall effect of this multistrategy closed-loop learning approach is (i) a significant increase in the performance of the learned concepts and (ii) a significant decrease in the complexity of the concept descriptions.

18.6 EXPERIMENTAL RESULTS

The experimental validation of the developed multistrategy approach to learning from engineering data and the verification of the expected effects were done using a typical engineering problem involving object recognition (through texture characteristics) for computer vision. The texture data used in our experiments was composed of the 12 texture classes presented in Figure 18.3. Different texture areas

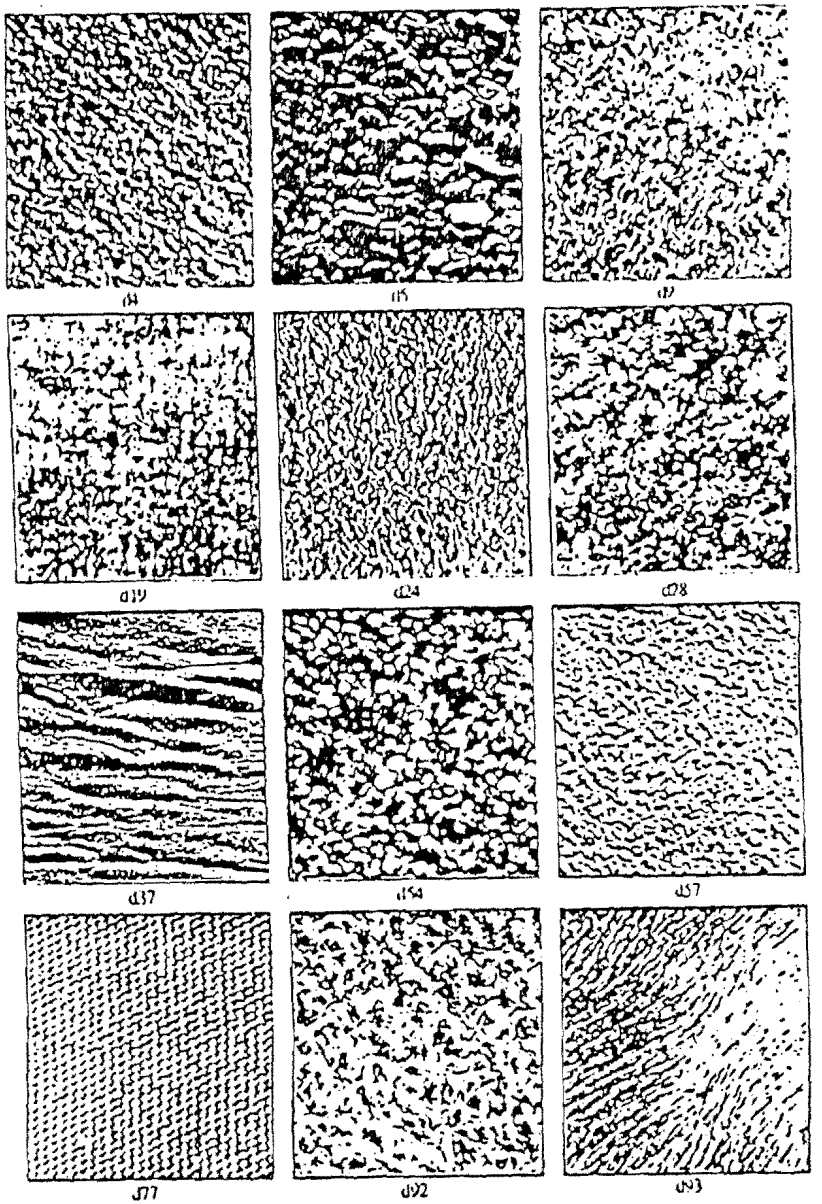

Figure 18.3: Texture images used in the experimental validation of the developed multistrategy learning approach (from Brodatz, 1966)

were provided for the learning and testing phases to secure an independent judgment of the experimental validation. Learning data (both training and tuning examples) were chosen randomly from indicated image areas. Testing examples for the recognition phase were extracted for the same 12 classes but from different image areas.

Examples of a given class were constructed using 8 attributes, and the attribute value extraction process was based on a modified version of Laws' method (Laws, 1980). In this method, texture attributes represent local texture energy at a given position on the image. The value of texture energy is calculated by convolving a special mask with an image and locally averaging received responses. The resulting attribute value is smoothed but still very irregular, complex, and highly noisy (Pachowicz and Bala, 1991).

The learning dataset was composed of 300 examples per class for 12 texture classes. The learning set was divided randomly into a group of training examples (200 examples per class) and a group of tuning examples (100 examples per class). An additional set of testing data of 200 examples per class was extracted from different image areas.

The testing phase was applied whenever (1) concept descriptions were exchanged between the two learning modules, and (2) concept descriptions were modified by the GA module. In this way, the effectiveness of all changes made to the ruleset was monitored continuously.

In the experiments reported here, the loop of inductive learning and genetic algorithms was run twice. The obtained experimental results are presented in Figure 18.4, in which white circles of the diagrams represent characteristics obtained for tuning data used to guide the genetic search, and the black circles represent characteristics for the testing data.

The experiments proceeded as follows. First, texture concepts were acquired by the AQ-NT program from 200 examples per class. Then, the GA module was activated to optimize the worst performing concept (in this case, the description of class d92). After 10 generations, the descriptions of all concepts were fed back to the inductive learning module, and the initial training dataset was filtered by the current set of concept descriptions. Examples that were not matched strictly by the rules were removed from the initial set. Examples that passed the matching criterion were forwarded to the next iteration of the inductive learning. After generating a new set of rules, the learned concepts were sent again to the GA module for further performance improvements.

The correct classification rate for class d92, obtained after two learning loops, was above 60%. That is a significant increase in comparison with 45% obtained from inductive learning only. The evaluation function (CC/MC) was defined to be the ratio of correct classifications to mis-classifications for all twelve texture classes. The CC/MC parameter was monitored for both tuning and testing data. An increase in CC/MC for the tuning data was observed. The CC/MC parameter for

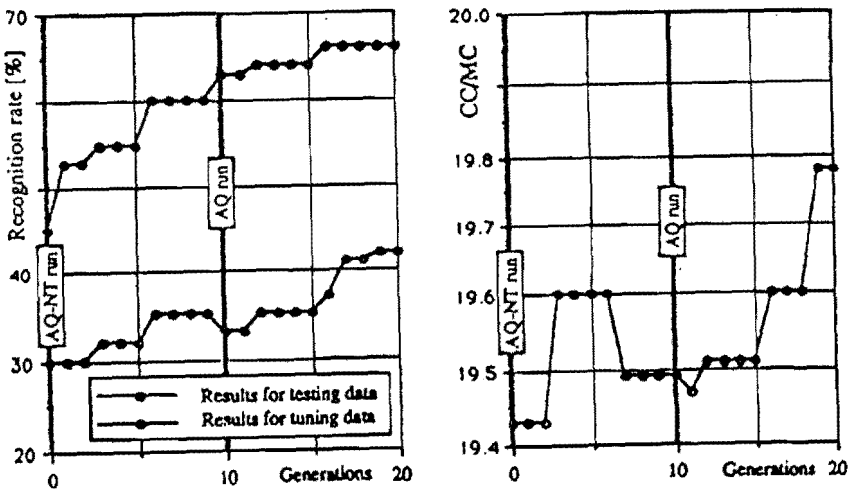

Figure 18.4: Experimental results for two runs of the learning loop integrating inductive generalization and genetic algorithms

testing data remained at the same level. All three parameters (the recognition rate of class d92, CC/MC for tuning data, and CC/MC for the testing data) indicate a significant overall improvement in system performance.

It was also pleasing to note that the number of disjuncts in the optimized class (d92) was reduced significantly from 36 disjuncts to 27 disjuncts, thus achieving an improvement in simplicity as well as performance.

The effects on system performance due to running the GA for a larger number of generations was also investigated. However, it appears that most of the improvement (both the decrease in complexity of the d92 class description and the increase in performance) is obtained in a very few number of generations.

18.7 CONCLUSIONS

A novel approach to the acquisition of concept descriptions from real-world engineering data has been presented. The approach is based on the idea of multistrategy learning, and the current system integrates symbolic inductive learning and genetic algorithms in a loosely coupled closed-loop fashion. The result is a robust concept learning system capable of learning simple, high performance descriptions from large, complex datasets with inherent noise.

Although there exist substantial differences between symbolic induction methods and genetic algorithms (learning algorithm, performance elements, knowl-

edge representation), their integration as presented in this chapter serves as a promising example that a better understanding of abilities of each approach can lead to novel and useful ways of combining them.

There are a number of directions for further research. The methodology must be tested on additional engineering problems presenting similar difficulties. More effective controls must be developed for the closed-loop cycles. The possibility of using constructive induction should be explored.

However, even in its present form the presented system is applicable to a wide range of difficult engineering problems.

ACKNOWLEDGMENT

This research was done in the GMU Center for Artificial Intelligence. Research of the Center is supported in part by the Defense Advanced Research Projects Agency under the grants administered by the Office of Naval Research, No. N00014-87-K-0874 and No. N00014-91-J-1854; in part by the Office of Naval Research under Grants No. N00014-88-K-0397, No. N00014-88-K-0226, and No. N00014-91-J-1351; and in part by the National Science Foundation Grant No. IRI-9020266.

References

Bala, J.W., DeJong, K., and Pachowicz, P.W. "Using Genetic Algorithms to Improve Performance of Classification Rules Produced by Symbolic Inductive Methods," in *Proceedings of the 6th International Symposium on Methodologies for Intelligence Systems*, Charlotte, NC, November, 1991(a).

———. "Learning Noise Tolerant Classification Procedures by Integrating Inductive Learning and Genetic Algorithms," in *Proceedings of the 1st International Workshop on Multistrategy Learning*, Harpers Ferry, WV, pp. 316–323, November 1991(b).

Bergadano, F., Matwin, S., Michalski, R.S., and Zhang, J. "Learning Two-Tiered Descriptions of Flexible Concepts: The POSEIDON System," *Machine Learning*, vol. 8, pp. 5–43, 1992.

Brodatz, P. *Textures: A Photographic Album for Artists and Designers*, Toronto, Dover Publishing, 1966.

Chien, S., Whitehall, B., Dietterich, T., Doyle, R., Falkenhainer, B., Garrett, J., and Lu, S. "Machine Learning in Engineering Automation," in *Proceedings of the 8th International Workshop on Machine Learning*, Evanston, IL, June 1991.

De Jong, K. "Learning with Genetic Algorithms: An Overview," in *Machine Learning*, vol. 3, no. 3, pp. 123–138, 1988.

Fitzpatrick, J.M., and Grefenstette, J. "Genetic Algorithms in Noisy Environments," in *Machine Learning*, vol. 3, pp. 101–12, 1988.

Holland, J. *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI, 1975.

Laws, K.I. "Textured Image Segmentation," Ph.D. Thesis, Dept. of Electrical Engineering, University of Southern California, Los Angeles, 1980.

Michalski, R.S. "AQVAL/1—Computer Implementation of a Variable-Valued Logic System VL1 and Examples of Its Application to Pattern Recognition," in *Proceedings of the 1st International Joint Conference on Pattern Recognition*, October 30, Washington DC, 1973.

———. "A Theory and Methodology of Inductive Learning," in *Machine Learning: An Artificial Intelligence Approach*, R. Michalski, J. Carbonell, and T. Mitchell (eds.), Tioga Publishing, Palo Alto, CA, pp 83–134, 1983.

Mozetic, I. 1985. "NEWGEM: Program for Learning from Examples. Program Documentation and User's Guide," *Report of the Computer Science Department*, University of Illinois at Urbana-Champaign, UIUCDCS-F-85-040, 1985.

Pachowicz, P.W., and Bala, J. 1991. "Improving Recognition Effectiveness of Noisy Texture Concepts through Optimization of Their Descriptions," in *Proceedings of the 8th International Workshop on Machine Learning*, Evanston, IL, pp. 625–629, June 1991.

——— "Texture Recognition Through Machine Learning and Concept Optimization," *Report of Machine Learning and Inference Laboratory*, George Mason University, MLI91-6, 1991.

Quinlan, J.R. "Induction of Decision Trees," in *Machine Learning* vol. 1, no. 1, Kluwer Academic Publishers, 1986.