Continuous Process Improvement Through Inductive and Analogical Learning

Pedro M. Saraiva and George Stephanopoulos
Laboratory for Intelligent Systems in Process Engineering, Dept. of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139

This article presents a methodology for the continuous detection and definition of process performance improvement opportunities, especially as these pertain to the quality of operations (such as product quality). The problem is first reduced to an essentially pattern recognition formulation, for which an integrated and adaptive methodology combining analogical reasoning and symbolic induction is developed. The resulting classification of past records of data is used to support the construction of a decision support system for the generation/selection of operating suggestions leading to performance improvements. The overall approach complements the usual set of statistical tools, commonly employed to address quality management problems. The basic methodology is also extended to handle fuzzy class definitions and function learning formulations. Case studies, covering both simulated and real industrial situations, illustrate the concepts and their practical utility.

Introduction

It has been broadly recognized that a *total quality* approach to the management of manufacturing and processing operations (taking into account production cost, consistency of product properties, safety, and environmental impact) is an essential precondition for the survival and growth of any company in the global market (Deming, 1986). Consequently, there has been a significant and continuously increasing awareness and interest on how the accumulation of operating data can lead to a better management of operational quality, which involves two complementary steps: (a) *control within prespecified limits* and (b) *continuous improvement of operational performance*.

The aim of the first is to detect "abnormal" situations, identify and eliminate the *special causes* that produced them. Once *control* has been achieved, the process reaches a predictable and stable random state of operation, characterized by a bounded variability (often designated as *capability*). But, getting the process under statistical control is not enough. The final level of variability achieved is the result of *common and sustained causes*, present within the process itself, and must not be considered as unavoidable. Instead, one should go one step further and move from *process control to process improvement*, that is, continuously search for common causes, ways of reducing their impact and challenge the current levels of performance (Juran, 1964).

Petroleum refining and petrochemical companies have been practicing this philosophy for some time. In those industries, dynamic models are employed to design, tune and deploy model-based controllers, which keep the operational variability within bounds determined by the magnitude of model-process mismatches. Static process models are also used to carry out linear or nonlinear optimization and thus determine the "best" levels of operating conditions (Lasdon and Baker, 1986; Garcia and Prett, 1986). Also, in many other processing and manufacturing systems (specially those with complex and/or poorly understood phenomena), *statistical process control* and *control charts* have been employed as maintenance tools, designed to keep the process under the current standards of performance. Designed experiments (Box, 1969; Taguchi, 1989; Ryan, 1989) are presently the main vehicle used to challenge these standards and thus achieve process improvement (such as factorial designs, orthogonal arrays, Evolutionary Operation).

To reduce the dispersion of the process output values and/or bring their average closer to a desired nominal target, we must usually search for better levels and ranges of operation

P. M. Saraiva is on leave from the Department of Chemical Engineering, University of Coimbra, Portugal.

for the manipulated variables. In order to do so, one must get a better understanding of the process and its behavior. Values of operating variables and performance indices, recorded and stored in a routine manner, constitute an excellent source of information to achieve increased process understanding. The implication is that efficient and competitive manufacturing is becoming more and more dependent on information management/processing activities. The amounts of data presently collected in the field are staggering; it is not unusual to find plants where the values of as many as 20,000 variables (Taylor, 1989) are being continuously monitored and accumulated. Unfortunately, in very few cases, if any, can it be claimed that all the potentially available knowledge is being extracted from the data. This lack of exploration and analysis is particularly severe for records collected while the processes were kept under "normal" operating contexts. Thus, most of the data generated are simply going to be "wasted" (less than 5% of the points contained in a conventional 6σ control chart are likely to attract any attention). Although some of the ideas and procedures that we will present are general enough to be potentially useful in other areas, such as fault diagnosis, the scope of applications that we are addressing concerns the analysis of data collected from a process that is under statistical control. The goal of our methodology is not to detect and diagnose "abnormal" situations, but rather to uncover improvement opportunities from the large amounts of "normal" operation data that are collected from the process but often end up not being explored at all.

For chemical processes where model-based controllers and model-based optimization of operating levels is a feasible and effective approach, collected data have been used to generate or update the process models. On the other hand, for manufacturing systems with poorly understood behavior, the economic losses due to the absence of systematic and continuous exploitation of accumulated operating data toward the achievement of process improvement can be quite significant. It has been estimated that only 6–20% of production problems are due to *special causes* (addressed by *statistical process control*), while the remaining bulk of 80–94% are due to *common* and *sustained causes* (Hart and Hart, 1989; Deming, 1986). Common causes can be approached only by *process improvement* studies and their impact reduced through the introduction of appropriate changes in the operating conditions and/or strategies.

The experimental nature of the techniques commonly followed to achieve process improvement has limited severely the *scope*, *frequency* and *number* of situations where such studies have been conducted (Hunter, 1989; Hahn and Dershowitz, 1974). Costs and risks associated with the execution of field experiments are the main reasons for that lack of implementations; one has yet to find the operator who likes the idea of having experiments performed in his/her production line. While some statisticians believe that experimentation is the only safe and reliable way to achieve process improvement, other statisticians and all the *empirical machine learning* researchers contend that by looking at past historical records and sets of examples, it is possible to extract and generate important and useful "new" knowledge. The history of science (Glymour, 1987; Shrager and Langley, 1990) has shown and our everyday experience seems to confirm that an *informed* and *systematic* observation of data, naturally generated, can indeed lead to the formulation of interesting and effective generalizations. The work described in this article is thus premised on the belief that *nonexperimental data analysis tools can uncover useful improvement opportunities at virtually no cost*. By avoiding the introduction of any sort of controlled perturbations in the production system, we will have to cope with such problems as: (a) masking of features; (b) the fact that some important variables will not seem to be so, because their values do not change in any significant way; and (c) increased difficulties in distinguishing between fortuitous empirical association and actual causal relationships. These assumed drawbacks, however, do not imply that significant improvements cannot be achieved. They simply restrict the space of possible improvements that can be suggested to the extent that they can be *induced* from the existing information.

We will present a methodology with the character of a *decision support system*, aimed at the *continuous detection* and *formulation of performance improvement opportunities* for processing and manufacturing systems that cannot be described effectively through first-principles models. The sole source of information to be used is a set of records containing the values of objectives and operating data, collected and accumulated on-line. As performance objectives we will primarily consider functions of product quality. Typical examples of such systems and objectives within the processing industries are the following: (i) the quality (evaluated by the kappa index) of the unbleached pulp sampled from the discharge stream of a continuous digester; (ii) the activity, yield, and total production cost of proteins from fermentation units; (iii) composition, molecular weight distribution, and structure of complex copolymers; (iv) product quality (as measured by the values of several independent physical characteristics) of fibers in a fiber-spinning plant; and (v) properties of silicon wafers produced in a chemical vapor deposition unit.

The methodology is based on *machine learning* approaches, integrating both an *instance-based* learning component and an *inductive symbolic* learning procedure that belongs to a group of techniques globally known as *top down induction of decision trees* (TDIDT). It is aimed to be complementary to the usual set of statistical tools that have been applied to solve problems as the above. Instead of just pointing out and diagnosing the existence of problems (as is the case with many statistical and connectionist techniques), the proposed approach searches and formulates *explicit operational suggestions to achieve process improvement*. The knowledge extracted from the operating data can lead either directly to the introduction of changes to the current operating conditions/strategies or to the design of a reduced set of confirmatory experiments, focused on the critical set of factors and levels identified. The overall learning methodology is also adaptive: it is able to update its output according to the additional "new" information that becomes available as more operating data are generated and collected on-line. This is accomplished through a *case-based reasoning* process, which is used to build and adjust the contents of an *active memory of exemplars*, upon which the inductive generalizations are performed.

Although many of the published industrial designed experiments consider only a single response, that is clearly insufficient in some situations where more than one performance variable must be taken into account. Extensions of the methodology described here to handle multiple operating objectives

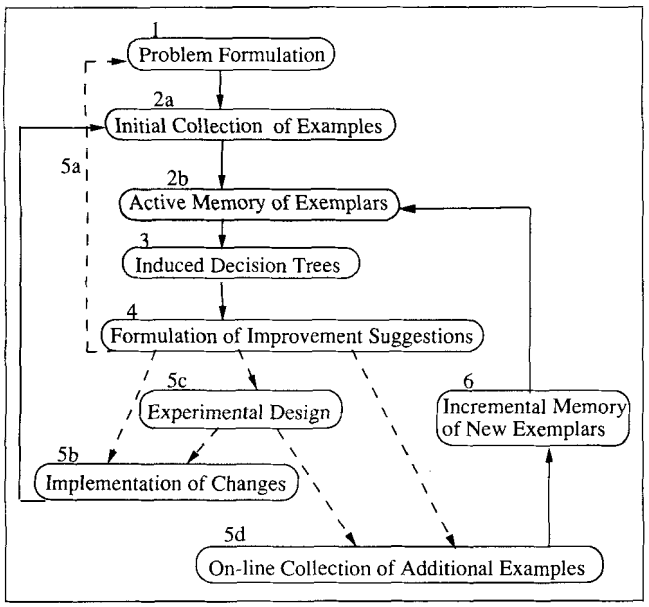


Figure 1. Global methodology.

simultaneously are in progress and will be described in a future publication. These extensions are centered around the adoption of a *blackboard architecture* (Nii, 1986); independent *cooperative agents* are associated with the multiple objectives, which interact within a *multicriteria decision-making* framework. Also, we will limit ourselves to the analysis of pseudo steady-state behavior situations, where the associated time scales are such that control-loop dynamics do not have to be taken into account. Accordingly, any integration between the present learning methodology and process control must be established at the supervisory level; we will search for windows in the operating space that lead to better performance, and this may result in a redefinition of the corresponding setpoints.

Subsequently, we will provide a general overview of the learning methodology, as well as a more detailed discussion of how decision trees are induced from a set of learning examples, the role of analogical reasoning in the definition and update of the active memory of exemplars, and how induced decision trees identify explicit process improvement suggestions. We will also discuss briefly extensions of the basic approach to handle fuzzy class definitions and function learning formulations of the quality problem (for more details, see Saraiva, 1991). Next, we will summarize the overall methodology through a final comprehensive functional diagram; then, results from four different applications (including both simulated and real industrial operating data) of the methodology will be used to illustrate the concepts described and their practical utility. Finally, we will establish some final conclusions and point directions for future work.

Overview of the Methodology and Its Characteristics

The methodology developed for the continuous improvement of process operations is composed of the following steps (Figure 1):

Step 1. Problem Formulation. Identify the variable y, used to evaluate process performance according to a particular functional characteristic (yield, product quality, and so on), and the corresponding measured operational variables or features $(x_1, x_2, \ldots, x_M)$, which are expected to influence the values of y.

Step 2. Collection and Labeling of Past Data. Using past operating records, a collection of pairs $[(x_1, x_2, \ldots, x_M); y]$ is constructed. Each of these pairs is labeled with the name of a particular class, according to its y value (for example, "good," "high," and "low"). A subset of critical examples, called the *active memory of exemplars*, is then identified and used to extract knowledge.

Step 3. Induction of Classification Decision Trees. The information contained in the active memory of exemplars is used to build classification decision trees. These decision trees define partitions of the space of operating vectors $\boldsymbol{x} = [x_1, x_2, \ldots, x_M]^T$. Such segmentation allows a new vector $\boldsymbol{x}$ to be assigned to one of the classes of y values, previously identified.

Step 4. Formulation of Performance Improvement Suggestions. The induced decision trees, constructed in step 3, contain a series of rules which indicate explicitly the ranges and levels that the operating features should have if the value of the performance variable, y, is to be mostly a member of a given class. These rules are submitted to several final procedures of *refinement*, *evaluation*, and *validation*, leading to the identification of a reduced subset of statistically significant feasible improvement opportunities, considered to be particularly "promising."

Step 5. Implementation of Improvement Suggestions. The reduced subset of final improvement opportunities identified in step 4 is analyzed by the decision-maker (such as process operator and production engineer). If feasible and validated, these suggestions can lead directly to implementation by introducing the corresponding changes into the process. Alternatively, the decision-maker may prefer to conduct a small set of designed experiments to confirm and explore specific suggestions before any implementation takes place. Lastly, in case none of the formulated improvement opportunities is considered to be interesting enough (or survives the tests performed in step 4), one can simply decide to keep collecting additional data and enrich the current *active memory of exemplars* with new operating vectors (step 6). It is thus hoped that as more data are generated with time, it will become possible to identify reliable and promising improvement suggestions.

Step 6. Evolutionary Sequence of Performance Improvement Suggestions. Through analogical reasoning, examples that convey additional important pieces of information can be identified on-line, as they become available. These new exemplars are then combined with others previously stored, resulting in an updated active memory of exemplars. This redefined memory supports the construction of new decision trees and thus leads to the generation of new performance improvement suggestions.

In the following, we will discuss the characteristics of the techniques employed to carry out the main tasks contained in the above six steps of the overall methodology. In later sections, we will provide a more technical and detailed description of the corresponding procedures.

Classification of operating data

Consider a process (possibly under feedback control) whose performance is being measured by variable y (such as a measure

of product quality). Let $[x_1, x_2, \ldots, x_M]^T$ be a vector of measured (and/or computable) *features*, which are assumed to have an impact on the values of y. These features may represent controlled outputs, manipulated inputs, measured external disturbances, violation of output constraints, operator's actions, and/or set-points. They may also include the values of quantities related to such measured signals as derivatives, integrals, and averages. Variables can be Boolean (for example, a particular valve being on or off), categorical (name of the operator in duty at the time the data were generated or the qualitative value of an operating variable such as "small" or "large"), or real (flows, temperatures, and derived quantities). Then, an *operating pattern*, $\boldsymbol{x} = [x_1, x_2, \ldots, x_M]^T$, is designated as any M-dimensional vector $\boldsymbol{x}$ defined in the feature space, S_x. To each collected pattern, $\boldsymbol{x}_i = [x_1, x_2, \ldots, x_M]_i^T$, corresponds a particular y_i value and the associated class of process behavior (y is quantized and considered to be "high," "low," "good," and so on).

The identification of the number of crisp classes (or fuzzy sets, in one of the methodology extensions that will be presented later), as well as the ranges of y values (or degree of membership functions) associated with them, may not be an easy task to accomplish. We will rely on the process experts and their specific knowledge for such definitions. A typical approach would involve the distribution statistics (average and standard deviation) of y, which can be explored to determine the ranges of y values to be associated with each of the classes (similar to the 2σ and 3σ limits in a control chart). However, if for some reason it is strongly felt that the quantification of y is undesirable, "unnatural," or particularly difficult to achieve, one can still employ the present methodology in terms of its function learning extension. In doing so, the definition of y classes becomes totally unnecessary, as one tries to study directly the mapping between operating patterns and continuous real y values. Similarly, for the identification of x_i variables associated with performance variable y, specific knowledge about the process or relationships among variables can be explored, and eventually combined with tools such as the KJ method, Ishikawa diagrams, or directed graphs of interactions.

Assuming that the initial problem formulation has been concluded and that K distinct classes ($C_1, C_2, \ldots, C_K$) were defined, *pattern classification* or *pattern recognition* is the process through which one tries to assign a given operating pattern $\boldsymbol{x}_i$ to the class to which it indeed belongs. In other words, pattern classification is a procedure through which the feature space S_x is partitioned into K mutually exclusive regions, $S_x^{(I)}$, $I = 1, \ldots, K$. Thus we have $S_x^{(I)} \cap S_x^{(J)} = \{0\}$, for $I, J = 1, 2, \ldots, K$ and $I \neq J$, while $S_x^{(1)} \cup S_x^{(2)} \cup \ldots \cup S_x^{(K)} = S_x$. Some of the most common approaches used to solve classification problems rely on the definition of K *discriminating functions*, $d_I(\boldsymbol{x})$. If $d_I(\boldsymbol{x}) > d_J(\boldsymbol{x})$, for all possible J, $J \neq I$, then pattern $\boldsymbol{x}$ is assigned to class I. Also, the decision boundaries between classes I and J are defined by the equation $d_I(\boldsymbol{x}) = d_J(\boldsymbol{x})$, provided that $d_I(\boldsymbol{x}) > d_L(\boldsymbol{x})$, for all $L \neq I, J$. Many techniques have been developed to solve classification problems. They can be divided into two groups, which we will designate as (a) *model-based* (or *parametric*) and (b) *memory-based* (or *nonparametric*) classifiers. In the first group, one postulates specific functional forms for the assumed discriminant functions (for example, linear, polynomial) and computes the corresponding parameters, or extracts the best from a superstructure of imbedded models (for example, neural networks) by minimizing a prespecified cost function. In any case, a partition of the feature space is explicitly defined, and the success of the classifier depends on the nature of the assumed model used to identify the decision boundaries, the characteristics of the cost function, and the accuracy of the estimated parameters. On the other hand, memory-based classifiers (nearest neighbors) use past data directly in the prediction scheme. New observations are classified by comparing them to similar examples previously generated and stored in memory (LaVigna, 1989). All these computations are done directly without the intermediate step of building explicitly a model of the feature space and are thus free from errors resulting from the choice of an "unreasonable" type of model. However, a proper measure of similarity among examples must be defined, and the accuracy of the classifier thus becomes dependent on the particular functional form and parameters used in that metric.

Performance improvement through classification of operating data

In this article, we are primarily concerned with the generation of suggestions to achieve process performance improvements. Although in the formulation of the overall strategy the classification problem is an important component and the information contained in past operating patterns (*examples*) is used to build a partition of the feature space, the global task itself is not a strict classification. In our methodology, classification is a means of detecting improvement opportunities, rather than an end by itself. It is used here to characterize records of data (alternative formulations will be described later), but to achieve process improvement one needs to be able to interpret and explain the data, as well as to generate operating suggestions on how to achieve performances better than the current one. Thus, the output of any classifier used within this context should be expressible in symbolic terms and amenable to an explicit interpretation and analysis by the decision-maker, as well as easily convertible into practical implementations. In this regard, all the conventional classification procedures are rather ineffective. For example, consider the curves $f_{1\text{-}2}$, and $f_{2\text{-}3}$, which characterize the boundaries between the classes C_1, C_2 and C_3 of y values (Figure 2), and provide a definition of the zones of the feature space that will lead to mostly class C_1, C_2, or C_3 operating patterns. An implicit classifier (statistical, neural network) obtains an approximation of the boundary surfaces $f_{1\text{-}2}(x_1, x_2)$ and $f_{2\text{-}3}(x_1, x_2)$. To determine operating patterns leading to class C_2 performance, one then needs to solve the following set of simultaneous inequalities:

$$f_{1\text{-}2}(x_1, x_2) \geq 0 \text{ and } f_{2\text{-}3}(x_1, x_2) \leq 0,$$

through an iterative procedure, such as the application of optimization techniques. But, such an approach yields single solution points and cannot define easily an explicit zone of operating conditions, as desired. Therefore, implicit classifiers are not well suited to answer questions such as: "*what are likely to be promising and interesting rectangular zones in the decision space, S_x, toward which one could move the current operating conditions to achieve better ratios of desirable vs. undesirable patterns*?" Although it is possible in a two-dimensional problem and by visual inspection, to identify such

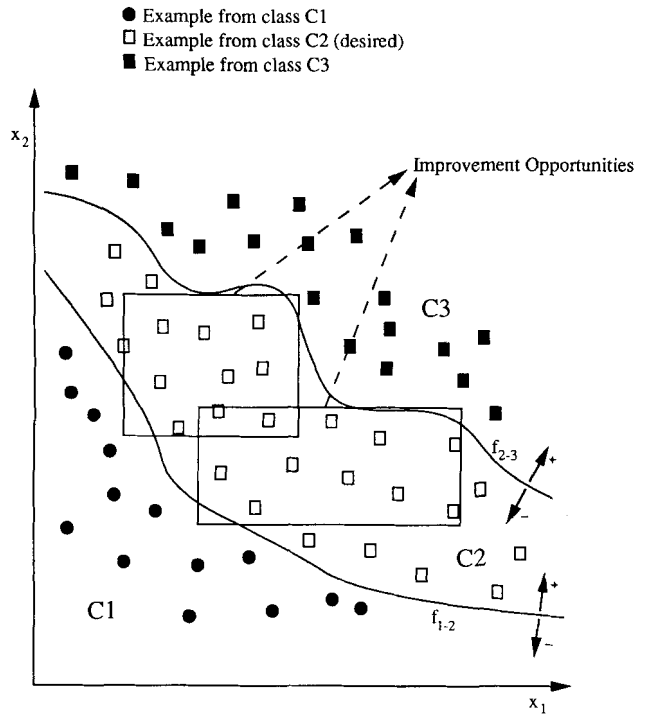


Figure 2. Process improvement and strict classification.

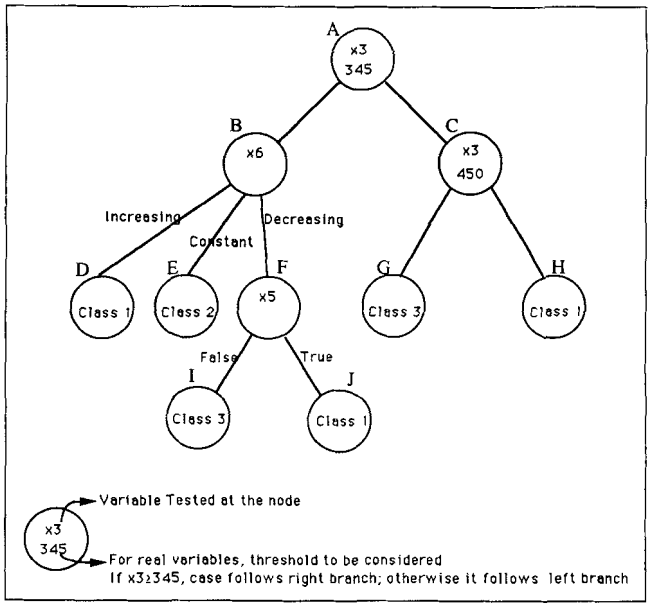


Figure 3. Example of a classification decision tree.

zones given the curves f_{1-2} and f_{2-3}, one would like to have a classification procedure that automatically identifies promising rectangular zones in the feature space (such as the ones presented in Figure 2) and able to do so even in higher dimension problems, where visual inspection is not possible. While all the conventional classifiers fail to answer promptly to the question raised above, in our methodology we employ a classification scheme that meets those requirements and is based on a *symbolic inductive learning* strategy, known as *top down induction of decision trees* (TDIDT).

Decision trees as symbolic classifiers

A decision tree is an intrinsically structural and sequential classifier. At the top node (A in Figure 3), a first test is performed, based on the value assumed by a particular feature (x_3 in the tree shown in Figure 3). Depending on the outcome of this test, a particular case with an associated measurement vector $\boldsymbol{x}$ is sent to one of the branches emanating from node A. There will be as many branches as the number of possible outcomes for the test performed. After this test has been completed and the case is assigned to a particular branch, a second test follows and is carried out at, for example, node B, over the values of some specific feature (x_6). This procedure is repeated until, after a last test, the vector $\boldsymbol{x}$ follows a branch that leads to a *terminal node* (or *leaf*), indicating the particular class, C_j, to which the specific incoming case is predicted to belong. For example, the following rule is associated with node J of Figure 3:

"*If* $x_3 < 345$ *and* x_6 is decreasing *and* x_5 is true,
then the corresponding y value belongs to class 1."

Rules such as this one can be easily used and understood by the decision-maker, and they identify explicit suggestions about operating pathways that would lead to given classes of y values.

Several additional properties make a TDIDT approach particularly attractive within our application context:

- No assumptions need to be made about population probability distributions.
- It can handle uniformly all types of operating features: Boolean, categorical, and/or real-valued.
- There is no need to scale the values of the several components of the $\boldsymbol{x}$ vectors, as the results obtained are invariant to any monotonic transformation of individual variables. For example, if x_1 is one of the features considered in the definition of $\boldsymbol{x}$, but the performance objective y actually depends on x_1^3, all the results obtained will still be valid ($x_1 > 3$ and $x_1^3 > 27$ perform exactly the same partition of cases).
- Decision trees provide a structured and hierarchical partition of the feature space.
- TDIDT-based classifiers have been successfully tested in quite a large number of real-world domains (Breiman, 1984; Muggleton, 1990; Hayes et al., 1985).
- Unlike many of the other classifiers (standard feedforward neural networks trained by backpropagation), very few *a priori* and somewhat arbitrary design decisions need to be made to build decision trees from a set of examples.
- Tests based on the values of irrelevant features are not likely to be present in the final induced decision tree, thus reducing automatically the problem dimensionality.
- Past numerical experience and empirical comparative studies have indicated that induced decision trees present accuracy levels competitive with any of the other alternative classification procedures (Breiman et al., 1984; Weiss and Kulikowski, 1991).
- Decision trees provide a modularized description of the feature space, exploring local information. Thus, they can capture adequately heterogeneous behaviors, and represent systems properly where different phenomena dominate in different regions of the feature space.

Active memory of exemplars

In the following section, we will discuss how the training process generates a sequence of decision trees from a set of (x, y) pairs of past operating data. While formal statistical inference is premised on the principle of random sampling, we will advocate here that certain pairs (x, y) are significantly more informative than others. Thus, the construction of decision trees can be supported by only this reduced subset of examples, which are situated in the feature space near the decision boundaries and constitute the *active memory of exemplars*. The on-line, posterior generation and collection of new examples will lead to a modification and update of this active memory, and consequently to an adaptation of the knowledge captured by the corresponding reinduced decision trees. In the subsequent section, we will discuss the procedures used to generate the initial active memory of exemplars, update it as new data become available and revise the induced knowledge captured by the decision trees.

Refinement, evaluation, validation, and implementation of extracted knowledge

As it is only possible to disprove all empirical learning and inductive generalizations, it is necessary to examine carefully and validate the extracted knowledge. The successive processes of refinement, evaluation, and validation of the improvement suggestions contained in the induced decision trees will be discussed later in detail. The rectangular zones of the feature space associated with some of the terminal nodes are readjusted, and the corresponding rules are tested for statistical significance. Also, different scores, measuring several properties of the individual hyperrectangles, are associated with each of the screened and refined improvement suggestions. The overall methodology was conceived within the scope of a *decision support system*. All the potentially useful improvement alternatives that survive the above treatment are to be examined by the decision-maker. Depending on the outcome of the analysis of the situation performed by the process experts, one of the following three courses of action can be adopted:

(1) Some of the captured improvement suggestions are statistically significant, validated, accepted, and selected by the decision-maker for immediate implementation. This ends one stage of the improvement process. As soon as the implementation starts, a new phase begins, by going back to step 2 of the methodology and collecting/labeling initial sets of data.

(2) Only a moderate degree of interest and trust is put by the experts in some chosen improvement opportunities, among all those captured by the methodology. It can thus be decided before any permanent implementation takes place that a series of focused and confirmatory designed experiments should be conducted to explore further the above opportunities. The identification of factors to be considered and their corresponding levels is guided by the induced tentative suggestions, reducing significantly the number, extent and cost of the experiments. When compared with the requirements of a conventional experimental approach to process improvement, if it were to be conducted from the beginning, one may omit the screening phase and center the attention around the zones of the feature space found to be particularly interesting by the learning methodology. As a rule-of-thumb, such an approach will reduce the total effort involved by 25–50% (Schmidt and Launsby, 1989). This type of guidance and ability to trigger designed experiments are generic and independent of the particular strategy adopted for conducting the experiments (fractional or full factorials, response surface analysis, Taguchi methods, etc.).

(3) No significant improvement opportunities were captured, and/or the support of process experts in favor of any of them is very weak. Under these circumstances, and before introducing any change into the process or doing any sort of experimentation, it can simply be decided that one should try to keep accumulating additional examples, and incrementally adapt and update the active memory of exemplars. When enough novelty appears to be present, a revision of the previously induced knowledge is attempted, this time hopefully leading to one of the other two situations, (1) or (2) above. This periodic procedure can be kept going for as long as needed, until enough evidence makes at least one particular improvement opportunity reliable and interesting enough to lead to either a direct implementation or experimental design.

The overall methodology was partially inspired by observing how process operators tend to articulate their reasoning activities in the control rooms of complex pieces of equipment, and it emulates to some extent the human problem-solving and learning process. The inspiration and influence of ideas and techniques from cognitive psychology/artificial intelligence can be found in a number of features of the methodology proposed in this article. The most important are as follows.

(a) There is an explicit concern about the vocabulary used to describe the induced knowledge, and how it fits for the ways people express themselves about the same problem domain.

(b) It can handle both qualitative and quantitative information.

(c) Some of the pioneering work associated with the development of algorithms to induce decision trees (Hunt, 1962, 1966) was done by experimental psychologists, who concluded that such algorithms are analogous in several aspects to human concept learning.

(d) Final improvement suggestions can be expressed as a set of rules, a particular knowledge representation format that has been shown to play a key role in people's reasoning activities.

(e) Current human learning theories for the process of concept acquisition point out the existence of a combination of template matching with stored exemplars, intensional generalized abstraction, and prototype formation. Similarly, our framework integrates both cases and analogical reasoning with induced generalizations, although it does not explore the identification of prototypes.

(f) The methodology uses raw examples as the basis of adaptive behavior, a process that is also behind peoples capabilities to adjust themselves to the surrounding environment (Riesbeck, 1989; Schank, 1984; 1988).

(g) The algorithms used to build and update active memories of exemplars are based on a number of memorization and learning sound principles from a cognitive psychology point of view (Anderson, 1990): *attention is a scarce resource, without which no storage takes place; learning is to a large extent an expectation failure and error driven process; repetition plays an important role with respect to the memorization and recalling of previous experiences*; and *near-miss examples convey critical pieces of information* (Winston, 1984).

Construction of Decision Trees from a Set of Examples

In the previous section, we discussed how the solution of a *classification, pattern recognition*, or *concept learning* problem constitutes one of the centerpieces of our approach to process improvement. In other words, to improve the process we are looking for factors and operating strategies (defined in terms of the features x_i, $i=1, 2, \ldots, M$), which will lead mostly to the desired class(es) of y values and/or to avoid the undesired one(s). These factors and strategies are contained in the extracted knowledge, represented explicitly in the form of decision trees induced from a set of data records. In this section, we will describe the technical details involved in the construction of such decision trees.

The process of induction of decision trees that we will present has been influenced by earlier works such as: (a) Quinlan's algorithms ID3 and C4 (Quinlan, 1986, 1987); (b) the CART procedure developed by Breiman et al. (1984); and (c) the sequential pattern recognition ideas and methods studied by Fu et al. (1968). It is composed of the following two tasks:

(1) Creation of a fully expanded tree, T_{max}.

(2) Pruning and simplification of T_{max} to obtain a sequence of increasingly simpler trees. A final pruned and reasonably sized tree, T^f, with greater statistical reliability and/or providing a good trade-off between accuracy and complexity, can be identified within the above sequence.

Our own implementation for this part of the methodology follows the ID3 approach for the creation of T_{max}, and the CART algorithm for the pruning phase.

Construction of T_{max}

As it can be implied from the decision tree of Figure 3, the construction of T_{max} from a *learning set of examples*, L_s, involves the following three types of decisions:

(1) Define and Select the tests to be made at each node.

(2) Select a criterion to stop the splitting process and consider a node as terminal.

(3) Label terminal nodes.

Definition and Selection of Tests. The construction starts at the root node, where all the learning examples are allocated. The definition of tests at the different nodes splits the learning set and sends subsets to the branches emanating from them. The space of possible tests that can be performed at any node is easily determined for binary and categorical variables. In both cases, identifying a test is equivalent to selecting the variable to be tested and associating one branch with each of its possible values. For real-valued attributes, besides the choice of a particular variable, we have to consider which threshold to use to define completely the split. The total number of possible threshold values that must be taken into account for each variable at a given node is limited. For a node t, containing $N(t)$ examples from the learning set, L_s, and a real variable, x_i, there will be at most $N(t)$ different values of x_i within t, which can be ordered as $x_{i,1} \le x_{i,2} \le \ldots \le x_{i,j} \le \ldots \le x_{i,N(t)}$. We define a corresponding set of values $\{z_{i,j}\}$, for $j=1, 2, \ldots, J$, with $J \le N(t)-1$, as follows:

$$z_{i,j} = \frac{x_{i,j} + x_{i,j+1}}{2}$$

These $z_{i,j}$ values are the only ones that must be considered as candidate thresholds for variable x_i, as any other possible threshold z in $(x_{i,j}, x_{i,j+1})$ will provide exactly the same split of the $N(t)$ examples as $z_{i,j}$.

Several specific criteria have been suggested for measuring the goodness of the particular split associated with one of the above tests. However, all of them have a common origin. Let s be a candidate test for node t, resulting in R possible outcomes, and denote as t_c, $c=1, 2, \ldots, R$, the children nodes emanating from node t according to s. Furthermore, let $N(t_c)$ be the number of examples that are allocated to the children node t_c. Then, one can construct a function Φ, which measures the "goodness" of the potential test s at the parent node t, as follows:

$$\Phi(s,t) = \phi(t) - \sum_{c=1}^{R} \frac{N(t_c)\phi(t_c)}{N(t)} \tag{1}$$

Different forms have been proposed for the function $\phi(t)$, which is a measure of the "impurity" of the set of $N(t)$ examples contained in node t. We will use a definition of $\phi(t)$ inspired by the information content measures derived initially by Shannon (Shannon and Weaver, 1964). The information associated with the occurrence of a given outcome for a probabilistic event is proportional to $\log(1/p)$, where p is the probability of that particular outcome. Thus, communicating that a highly likely alternative indeed took place does not convey a significant amount of information, while the opposite is true for a very unexpected or surprising outcome. Consequently, one can consider the average information associated with the allocation of an example contained in node t to a particular class as the weighted average of the information associated with its allocation to each of the K classes. Let $N_k(t)$, $k=1, 2, \ldots, K$, be the number of examples, from L_s and allocated to node t, which belong to class k. Then, $P_k(t) = [N_k(t)/N(t)]$, $k=1, 2, \ldots, K$, denotes the relative frequency of examples that belong to class k, among all the $N(t)$ examples contained in t. The function $\phi(t)$ can thus be defined as:

$$\phi(t) = -\sum_{k=1}^{K} P_k(t) \cdot \log_2 P_k(t) \tag{2}$$

There is a clear analogy between this function $\phi(t)$ and the statistical mechanics definition of entropy, as expressed by the Boltzmann equation (Tribus, 1961). A higher average information gain associated with the knowledge of the outcome for a probabilistic event corresponds to a higher initial degree of uncertainty or ignorance about that event. Thus, a dual interpretation can be given to $\phi(t)$, which also measures the "disorder" or "entropy" of the set of examples contained in node t. Figure 4 shows three different possible compositions for a node with a fixed total number of cases, illustrating how our intuitive notion of disorder is captured by the $\phi(t)$ function.

From Eqs. 1 and 2 we can derive the final criterion that will be used to select the tests that will be performed at each of the nodes of the decision tree:

$$\Phi(s,t) = -\sum_{k=1}^{K} P_k(t) \cdot \log_2 P_k(t) + \sum_{c=1}^{R} \frac{N(t_c)}{N(t)} \sum_{k=1}^{K} P_k(t_c) \cdot \log_2 P_k(t_c) \tag{3}$$

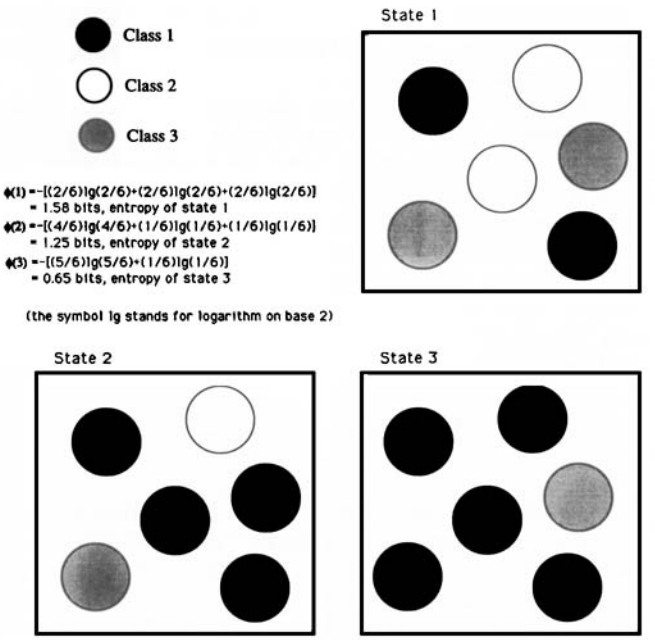


Figure 4. Sequence of three possible states for a node with 6 cases and corresponding entropy evaluations.

Note that $P_k(t_c)$ denotes the relative frequency of class k examples, among all the $N(t_c)$ cases allocated to the child node t_c, as a result of split s. By maximizing $\Phi(s,t)$ over the space of all possible splits, we will identify the particular test that results in the highest information gain. This test is considered to provide the best allocation of examples through the resulting children nodes. Thus, at each node (starting at the root), the search process attempts to identify the operating feature, x_i, and the associated split, s, that reduce the most the amount of remaining ambiguity, remitting decisions of lesser importance to be made in subsequent nodes. Both experimental and theoretical communication results have shown that using $\Phi(s,t)$ as objective function leads to the creation of compact and quasi-optimal-induced decision trees (Moret, 1982; Utgoff, 1988; Goodman and Smyth, 1990).

Criteria to Stop the Splitting Process. For mutually exclusive classes, nodes may be considered not to be terminal nodes unless they are completely "pure" (contain only examples that belong to a single particular class). Alternatively, a node t can be considered as terminal if one class is clearly predominant within it, or if t contains only a total number of examples which is believed not to be representative enough to deserve further division.

Labeling of the Terminal Nodes. With mutually exclusive classes and pure leaves, assigning a label to any of the terminal nodes is a trivial task, as they contain only cases from a particular class, that is, the label of the leaf becomes simply the name of the corresponding class.

For terminal nodes that are "impure" to a certain extent, two approaches can be followed for the labeling:

i) Applying the plurality rule, node t is simply labeled with the name of the most probable class within it. This is the approach we will usually follow when the problem has been formulated under the assumption of crisp class definitions (other alternative formulations will be enumerated later).

ii) Instead of making a definite assignment for the cases falling into node t, the decision tree acts as a probabilistic classifier, and each terminal node is labeled with the corresponding set of conditional probabilities, $\{P(k/t)\}$, $k=1, 2, \ldots, K$. It should be remembered, however, that here we are talking about the conditional probabilities *for an arbitrary case drawn from the process* (and not just from the learning set of examples, L_s) to belong to class k, given that it is assigned to the leaf t. Usually we will not employ simply a random sample of examples collected from the process as our learning set, but rather the active memory of "screened" exemplars. To get the above estimates for the underlying process population, we will use the induced decision tree to classify all the examples contained in a separate set of randomly collected instances (not used to build the decision trees), L_{tu}, designated as *tuning set*. If we let $N_{tu}(t)$ be the number of cases from L_{tu} placed at node t, and $N_{k,tu}(t)$ the number of examples among them that belong to class k, we can define the following as estimates for the desired conditional probabilities:

$$P(k/t)=\frac{N_{k,tu}(t)}{N_{tu}(t)} \tag{4}$$

Pruning of $T_{\max}$

The construction of the lower levels of $T_{\max}$ is based on the information provided by a progressively smaller number of examples contained in the learning set, and the resulting nodes may not capture any significant problem structure or add important predictive power to the induced decision tree. Such *overfitting*, a problem also found in many other types of classifiers, when coupled with the presence of noise in the values of x or y, may result not only in an unnecessarily complex, but also inaccurate (when doing prediction) $T_{\max}$. To overcome these problems, several $T_{\max}$ pruning strategies have been developed. Among the available pruning methods, we have been using the *error complexity* procedure included in the CART algorithm (Breiman et al., 1984). This procedure builds a sequence of gradually simpler trees by local detection and removal of unimportant nodes.

Let T_t be the subtree having as its root the node t of a mother decision tree, T, and $T-T_t$ the remaining tree that one obtains from T after removing T_t from it, and thus making t a terminal node. The notation $T_n > T_m$ will be used to indicate that T_m is a pruned version of a larger tree T_n. The first step of the pruning process consists of building a sequence, $\{T_i\}$, $i=1, 2, \ldots, F$, of pruned versions of $T_{\max}$, such that $T_{\max} \equiv T_1 > T_2 > \ldots > T_i > \ldots > T_F \equiv \{t_1\}$. The final tree in the sequence, T_F, is a degenerate tree, composed of just one node, the root node of $T_{\max}$, t_1. To find $\{T_i\}$, so that it represents an effective generalization of the induced knowledge at different levels of resolution, the inherent trade-off between accuracy and complexity is addressed explicitly. Consider a particular subtree, T_t, and let $R(T_t)$ be the ratio of (a) the total number of examples contained in L_s that are assigned to and misclassified at any of the leaves of T_t, and (b) the total number of examples present in the learning set, L_s. Thus, $R(T_t)$ is a measure of the lack of accuracy associated with T_t. On the other hand, the complexity of T_t is associated with the total number of leaves of the subtree and will be denoted as $C(T_t)$. An accuracy-complexity measure can then be identified with the following weighted summation of $R(T_t)$ and $C(T_t)$:

$$RC_\alpha(T_t) = R(T_t) + \alpha C(T_t) \quad (5)$$

Clearly, if the subtree T_t were to be pruned and the note t considered as terminal, Eq. 4 would yield

$$RC_\alpha(t) = R(t) + \alpha \quad (6)$$

since $C(t) = 1$.

Furthermore, for any subtree where the equality $RC_\alpha(T_t) = RC_\alpha(t)$ holds for a small and positive α value, the justification for keeping it in the initial decision tree is quite weak. The resulting values of α for each subtree,

$$\alpha^*(T_t) = \frac{R(t) - R(T_t)}{C(T_t) - 1} \quad (7)$$

thus represent a localized measure of the intrinsic merit of subtree T_t and can be interpreted as the average increase in accuracy associated with each of the subtree's leaves. At any stage of the construction of the sequence $\{T_i\}$, starting with T_{max}, the $\alpha^*(T_t)$ scores are evaluated for all possible subtrees, and the particular one with the lowest value is considered to have the "*weakest link*" to the mother tree and is pruned from it, leading to the definition of the next member in the above sequence.

Breiman et al. (1984) have proved theoretically the *existence* and *uniqueness* of $\{T_i\}$, generated in the manner described above, as well as the fact that the corresponding sequence of $\alpha^*(T_t)$ values, $\{\alpha^*_i\}$, $i = 1, 2, .., F$, is monotonically increasing, starting with $\alpha^*_1 = 0$.

By evaluating the classification accuracy of the several induced decision trees contained in $\{T_i\}$ when predicting the classes for all the members of the tuning set, L_{tu}, one of them can be identified as a reasonably sized and *final induced decision tree, T^f*. This choice can be determined either by (a) interaction with the user or (b) considering as final induced decision tree the simplest tree in the sequence whose error rate on the tuning set is not significantly higher than the lowest value achieved by any of the other members of $\{T_i\}$ (Saraiva, 1991; Breiman et al., 1984). In all the case studies, we have followed the second criterion.

Computational complexity

TDIDT techniques are among the most efficient inductive learning algorithms. The computational load grows only in an approximately linear way with the problem complexity. As all sequential classifiers, decision trees are also quite fast at running time, since only the particular path of tests leading to a given leaf needs to be performed to predict the class of a new incoming vector $\boldsymbol{x}$.

From a geometric point of view, it is useful to recall that TDIDT algorithms define recursively a division of the S_x space into hyperrectangles. Such kind of partitioning has been proven to be computationally optimal for several learning tasks with respect to the number of examples needed to make accurate predictions with high confidence (Salzberg, 1990). Rectangles have also been commonly used to approximate other shapes and are particularly economical to store and manipulate (Samet, 1988).

The time complexity (TC) of TDIDT algorithms has been quantified in several different, but equivalent, ways. Results refer only to the construction of T_{max}, since the pruning process imposes a very low additional load to the overall allocation of computational resources. Calling d the final depth (maximum number of tests needed to reach any of the terminal nodes) of the binary decision tree T_{max}, induced from a total of N_s learning examples, the time complexity of the algorithm includes $O(N_s M d)$ evaluations of the function $\Phi(s,t)$ and a sorting operation at each node, leading in the worst case to an overall computational load (Saraiva, 1991; Breiman et al., 1984) of order $O\{N_s M d [lgN_s - (d-1)/2]\}$.

It can thus be seen that the total computational effort is approximately only linear in each of the M, N_s and d parameters. Empirical studies have also confirmed this assertion.

A framework for the computational analysis of learning algorithms that is becoming increasingly popular is known as probably approximately correct (PAC) learning (Kearns, 1990). A PAC analysis of the induction of binary decision trees was also performed by Saraiva (1991).

Active Memory of Exemplars and its Adaptation

As described in the previous sections, our methodology tries to find regularities and extract useful knowledge from records of operating data. When provided with a batch of examples, such as the one presented in Figure 5, for a pattern recognition problem formulation, the complete set of available cases may be used for the induction of decision trees. However, a reduced subset of carefully screened exemplars usually conveys almost all the information needed, thus making the remaining examples become redundant. For instance, the information required to identify correctly the discriminating curves shown in Figure 5 resides in those selective cases that lie close to the decision boundaries, as illustrated.

Furthermore, for on-line applications it becomes intractable to simply keep adding indiscriminately new points to the memory. Not only would this result in the creation of storage and processing problems due to computational limitations, but also in the preservation of obsolete or noisy examples. To handle these issues, we incorporated in our learning framework the capability to build and incrementally update an *active memory of exemplars, L_{am}*, that is going to be used as the learning set from which generalizations are induced, following the steps presented in the sections on "Construction of Decision Trees" and "From Decision Trees to Operating Strategies." Within the context of permanent and on-line applications of our learning framework, one would like to be able to detect novelty and accordingly improve/revise the contents of L_{am}, given that additional "fresh" data are being collected from the process. As more examples become available, we should be able to update our dynamic memory for any of the following reasons:

1. New examples may cover different regions in S_x or provide greater relevant detail over regions of the feature space where some information already exists.
2. The active memory of exemplars needs to be updated to reflect drifts or other temporal changes of the process behavior. In that regard, some cases previously stored in L_{am} may become obsolete, thus contributing to a distorted portrait of the system that will also be reflected in the corresponding induced generalizations. These particular parasite examples should be iden-

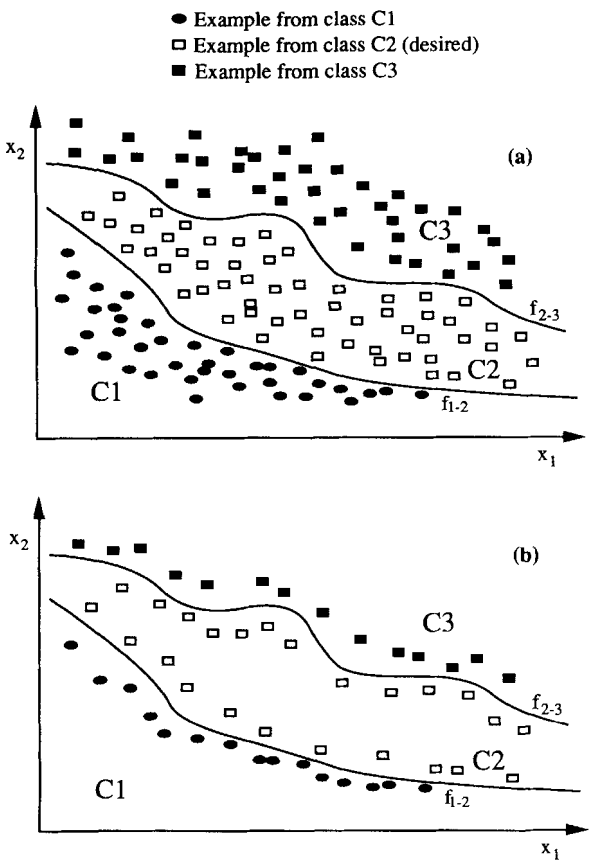


Figure 5. (a) Training set of examples and (b) screened memory of exemplars.

tified, excluded from memory, and replaced with new incoming and representative "fresh" data.

3. Noisy or unreliable exemplars need to be removed from L_{am}, as they also contribute to a distorted description of the actual process behavior.

Although this continuous definition and update of a dynamic memory of cases is integrated with the remaining parts of the methodology, it can be used in other contexts by itself, because it is a completely autonomous process. One may also look at it as a broader scope algorithm for adaptive storage of informative, nonredundant and critical data, able to react to any process modifications, and accordingly adjust the memory contents. This compressed and dynamic memory can then be used to support other data analysis approaches, such as neural networks training and building linear discriminant functions.

The next paragraphs will describe how the active memory of exemplars is constructed and continuously updated.

Let x_1 and x_2 be two arbitrary measurement vectors of examples contained in an initial randomly collected *training set*, L_{tr}, of N_{tr} examples. A normalized and weighted euclidean distance, D, between patterns is given by:

$$D(x_1,x_2)=\sqrt{\sum_{i=1}^{M} w_i^2 \frac{(x_{i,1}-x_{i,2})^2}{(x_{i,\max}-x_{i,\min})^2}} \qquad (8)$$

where

$$x_{i,\max}=\text{Max}\{x_{i,j},\ j=1,\ldots,N_{tr}\}$$

$$x_{i,\min}=\text{Min}\{x_{i,j},\ j=1,..,N_{tr}\}$$

The initial set of weights w_i, $i=1,2,\ldots,M$, is the solution of an optimization problem, the objective being the minimization of the error rate (fraction of examples misclassified) obtained when a *1-Nearest Neighbor* classifier using all the members of L_{tr} attempts to predict the classes of the examples contained in a *tuning set*, L_{tu}. Having thus defined an adequate distance measure, D, we proceed by identifying the *Tomek links* present within the members of L_{tr}. A Tomek link (Tomek, 1976; Park and Sklansky, 1990) is any pair of examples, with operating patterns x_1 and x_2 and belonging to different classes, for example, C_1 and C_2, respectively, which satisfy the following conditions:

Given the vector $z=\dfrac{x_1+x_2}{2}$

(a) $D(z,x_1)=\text{Min}\ D(z,x_i)$, over all the examples (x_i,y_i) in L_{tr} such that $y_i \in C_1$

(b) $D(z,x_2)=\text{Min}\ D(z,x_i)$, over all the examples (x_i,y_i) in L_{tr} such that $y_i \notin C_1$

Vectors x that belong to a Tomek link are near-misses placed in the feature space close to the decision surfaces that separate the members of two distinct classes. Only those examples from L_{tr}, which form Tomek links, are included in the initial *active memory of exemplars*, L_{am}, and will thus be used for the construction of the first generation of decision trees.

A *measure of reliability* for each of the exemplars kept in memory is determined and continuously updated by using a 1-nearest neighbor classifier (that employs the D distance measure) supported by all the members of L_{am}. With this classifier (which we will designate as DNN), we predict the classes of the examples contained in the current tuning set, L_{tu}, as well as of any new examples that are being generated and collected on-line from the process. As the measure of reliability associated with each individual exemplar $i(MR_i)$, we will consider the fraction of correct predictions achieved over all the times that specific case was found to be the closest neighbor and thus used to classify other examples. We will only keep in the active memory of exemplars those which are found to maintain a measure of reliability higher than a prespecified threshold, γ(for example, $\gamma=0.7$). Any exemplar i can thus be automatically excluded from L_{am} at time t, provided that the responsibility for the misclassification of the new incoming case, collected from the process at instant t, is assigned to that exemplar i and brings down its MR_i score to less than γ. When and if that happens, exemplar i is considered not to be any longer useful enough to deserve its current space in the active memory; consequently, it will be forgotten forever.

In addition to excluding cases with low MR_i scores, we are also going to build an *incremental memory of new exemplars*, L_{inc}. As new cases are collected on-line and become available, we will include in L_{inc} all those that were misclassified by DNN. This prediction failure is interpreted as a signal pointing out that the incoming data convey novelty and useful information.

The growth rate of L_{inc} (as measured by the number of new incorporated exemplars per hour of operation or any other similar scale) and the error rate obtained by DNN over a tuning set of cases are quite important indexes by themselves. If a process keeps reproducing its behavior over time, their values are expected to decrease gradually until they eventually reach

an almost constant value. On the other hand, if some important change takes place, they will have a sudden increase. An analysis of the temporal evolution of these values can thus be enlightening and explored as a diagnostic tool. We will examine some specific examples of these types of behavior later.

When enough new information and evidence have been collected, we are going to modify substantially the contents of L_{am}, and define a revised memory. Whenever the total number of exemplars contained in L_{inc} reaches a given fraction, β (for example, $\beta = 0.25$), of the present cardinality of L_{am}, we say that a *revision point* has been reached and initiate the corresponding *revision process*.

To improve the contents of the current active memory through the revision process, we first need to evaluate the present intrinsic merit of each of the individual exemplars contained in it. For that purpose, a *global measure of significance*, S_i, associated with exemplar i of L_{am}, will be considered. This metric takes into account and combines the following three factors: *reliability, frequency of use*, and "*age*" of exemplar i. To establish the S_i scores, we must keep track of the points in time at which the examples were generated and collected from the process. Furthermore, consider that in every occasion exemplar i was used by the DNN classifier to predict the membership of a new incoming case, it was *either* (a) rewarded with $a+1$ credit if the prediction happened to be correct *or* (b) penalized with $a-1$ payoff, if a misclassification took place. Then, S_i is simply a weighted sum of the rewards/penalizations that exemplar i received since it was added to L_{am}. The weights used to compute the above summation consider a linear increase of importance as time approaches the present. Thus, the reward/penalization given to a particular case, due to its use by the DNN to classify the example most recently collected from the process, is premultiplied by a weight of $+1$. On the other hand, the similar reward/penalization associated with the classification of the *oldest example* is premultiplied by a 0 weight: the performance over cases generated before this oldest example is not taken into account for the computation of S_i scores.

To identify what we designated as the oldest example, the value of a *time horizon* parameter, *th* (for example, $th = 3$ hours), must be specified. It leads to the definition of a time window, starting at instant tth and ending at the present time, t. The width of this window should be large enough for one to explore it and get enough information to evaluate the recent performance of each exemplar. It should also be small enough to exclude from consideration past system behavior that has very little to do with the current situation. Once a value has been assigned to *th*, our oldest example can be simply stated as the first case to have been generated by the process after the time instant tth. The weights associated with the reward/penalization for any of the intermediate cases (generated after tth and before t) lie in (0,1) and are found by linear interpolation, taking into account the time instant they were collected from the process, as well as the corresponding time tags for the youngest/oldest patterns.

When a revision point is reached, the global measures of significance for all the exemplars that belong to L_{am} are updated. Those not having positive S_i scores are considered not to provide any longer a valid contribution to the active memory, L_{am}, and thus are removed from it. This exclusion rule is complementary to the one based on the values of the measures of reliability, MR_i; it withdraws from memory cases that used to be quite accurate and frequently employed, but became suddenly obsolete due to shifts in the process behavior. Given their past cumulative history, it would take a very long time for the MR_i scores of such cases to decrease significantly. However, less than about one time horizon needs to be elapsed after any process change occurred for the S_i values to become nonpositive.

As all the new exemplars contained in L_{inc} are believed to be potentially useful and rich in "fresh" information content at each revision point, they are added to L_{am}, while leaving L_{inc} empty and ready to restart the process of finding and storing new significant incoming exemplars. Finally, only those members of the already redefined memory that belong to at least one Tomek link are preserved. These two operations conclude the revision process and thus the definition of a renewed active memory, from which another generation of induced generalizations can be constructed.

This adaptive process can be repeated again and again, for a number of times, or even on a never-ending basis for permanent on-line implementations of the learning methodology. For the latter type of situation where many revision points are reached, after a number of cycles, the size of the active memory could eventually become too big, mainly in problems with a feature space of high dimensionality. If that is the case (not observed in any of the studies that we have conducted so far), a maximum acceptable cardinality can be defined to avoid this prohibitive growth. To prevent exceeding that limit, we will withdraw from L_{am} as many exemplars as necessary, which were contained in the current active memory after the usual revision process had been completed, and which (a) were already members of L_{am} before the revision point was reached and (b) have the lowest (although positive) S_i scores among all the cases that verify (a).

After revising and updating L_{am}, as was just described, a new tuning set of examples is formed. This tuning set consists of the N_{tu} consecutive cases collected from the process either (a) immediately before the revision point was reached or (b) immediately after the revision process has been concluded. Using this tuning set and the updated active memory, the distance measure $D(\boldsymbol{x}_1, \boldsymbol{x}_2)$ and MR_i scores are adjusted. A new compiled description of the feature space can also be induced from the renovated, active memory contents, thus leading to the construction of another generation of generalizations.

From Decision Trees to Operating Strategies

The terminal nodes of a classification decision tree identify a partition of the feature space, S_x, which is segmented into hyperrectangles associated with the different classes. The adoption of hyperrectangles as the basic primitives to describe S_x was motivated by observing that people in the control rooms tend to articulate their reasoning as *orthogonal thinking*: they express themselves by means of conjunctions of statements about the individual operating features, x_i (for example, the temperature is high, *and* we are currently using raw material A). Note that this language also allows one to address explicitly the consideration of hard constraints both in the performance and feature space, provided that they are expressed as equalities/inequalities on y (leading to the definition of "forbidden" classes of y values) or x_i (leading to the refinement of induced

hyperrectangles with the exclusion of infeasible parts of them). As the knowledge captured by the induced decision trees is already expressed in an adequate "vocabulary," its conversion into a set of useful operating strategies becomes an almost trivial translation exercise. Each terminal node of the current *final induced decision tree*, T^f, identifies a particular hyperrectangle defined in S_x that will be successively (a) *refined*, (b) *evaluated*, and (c) *validated*.

Refinement

Ideally, we would like to find hyperrectangles in the feature space not only as big as possible, but also as "pure" as possible. In other words, we wish to define and enlarge a rectangle from class C_I as much as we can, but only up to the point where further enlargement may be achieved without lowering significantly the fraction of C_I examples contained inside its volume. This trade-off provides a dual interpretation for the pruning process described earlier, and we will also use it explicitly here to guide the refinement of the boundaries associated with some of the T^f leaves.

As the starting point, we have the modularized description of S_x provided by T^f. Let's focus the discussion to a particular terminal node of T^f, t, labeled as belonging to class C_I. We will examine each of the surface boundaries of node t in its own turn. Some of these boundaries may establish borders exclusively with zones of the feature space considered to be also of class C_I, according to the partition defined by T^f (Figure 6a). The initial node t is expanded by moving each of the above boundaries (while keeping the others fixed), until a point is reached where further expansion would start including non-C_I space (once again according to T^f) into the modified version of t. Let's designate as $V(t)$ the normalized volume of a generic hyperrectangle. After scaling all the real attributes x_i to lie in [0,1], $V(t)$ is simply the product of the ranges (evaluated from the scaled x_i features) of values that are associated with the definition of t. Then, the location of the particular boundary whose expansion resulted in the modified hyperrectangle with the highest normalized volume is changed permanently. This expanded version replaces t, and the process is repeated having it as the new starting point. The iterative process continues, until no further expansions within class C_I space are possible (Figure 6b). After the conclusion of these expansion cycles for each of the terminal nodes of T^f, we obtain an identical number of eventually modified and partially overlapped hyperrectangles.

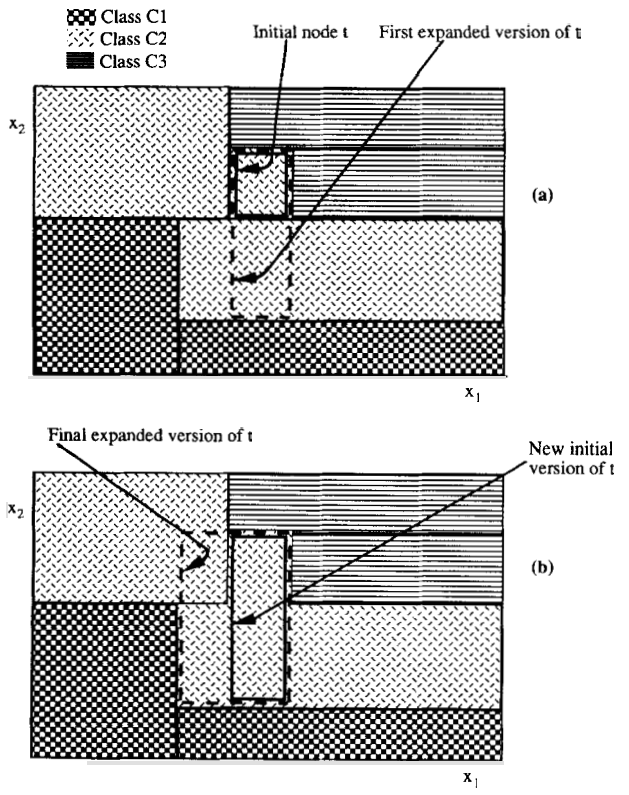


Figure 6. (a) Original partition of the feature space and first expansion of terminal node *t* and (b) final expanded version of the initial node *t*.

A final statistical test of significance and simplification is conducted on each of these modified hyperrectangles. Let's assume that one of them is defined by the following rule:

If P_1 and ... and P_i ... and P_p, *then* $\boldsymbol{x}$ belongs to class C_I,

where each P_i precondition represents a test performed on one of the original features x_i (for example, $x_i \in [a,b]$). To evaluate the statistical reliability of the lefthand side preconditions, P_i, each of them is picked up individually and tested against the null hypothesis of that particular precondition to be irrelevant for classification purposes. Either a χ^2 or an exact test of independence may be employed for that purpose (Saraiva, 1991), and only those preconditions, for which the null hypothesis can be rejected at a significance level less than or equal to a prespecified threshold (0.01), are retained in the definition of each of the hyperrectangles.

Also, any hard constraints established in the feature space as inequalities on x_i variables that intercept existing suggestions may be considered at this point. Only the feasible parts of such hyperrectangles (themselves also hyperrectangles) will be retained, thus leading once again to a final adjustment of some suggestions.

By the end of this third testing and refinement procedure, a group of improved, statistically significant, and simplified hyperrectangles are obtained, where t^* is the final revised version derived from the original leaf t contained in T^f.

Evaluation

Each of the t^* hyperrectangles found so far and the decision paths associated with them must now be analyzed and studied carefully. Different criteria will be used to characterize those hyperrectangles. A first evaluation index, called *certainty factor*, $H1(t^*)$, is the fraction of those examples from the tuning set and contained inside t^*, which indeed belong to class C_I. Thus, $H1(t^*)$ is an estimate of the conditional probability that a pattern vector $\boldsymbol{x}$ is associated with class C_I, given that it lies inside the zone of the feature space defined by t^*: $H1(t^*) = P(C_I/t^*)$. It provides a measure of the reliability and accuracy of t^*. A second evaluation criterion, designated as *Pareto Index*, $H2(t^*)$, allows one to identify those regions of the feature space where most of the examples from each class are currently being generated. Given the hyperrectangle t^*, $H2(t^*)$ is simply the fraction of the class C_I examples from the tuning set that are contained inside t^*. As a third index, we

will consider the *volume-reliability trade-off* score, $H3(t^*)$. It is a global evaluation criterion, given by the following linear combination of the normalized volume and certainty factor: $H3(t^*) = \tau V(t^*) + H1(t^*)$, where τ (for example, 0.2) is a constant that establishes the relative importance of both criteria. Finally, as a complementary fourth evaluation measure, we will consider the *demographic density*, $H4(t^*)$, which is the ratio of the total number of examples in the tuning set that belong to t^* and its normalized volume, $V(t^*)$.

Validation

All our methodology was conceived within the spirit of a *decision support system*. It is aimed at uncovering promising and feasible pathways to process improvement, and presenting this distilled information to the plant personnel. It is their responsibility to analyze the output and define the course of action to follow. To further help him/her in that process, the decision-maker can count on: the ability to visualize the final induced decision tree, T^f, or any other member of the $\{T_i\}$ sequence; the values of the different evaluation scores, $H1$ to $H4$, for all the t^* regions of the feature space and the capability to examine the corresponding rules that identify each of them; and the possibility of visualizing the projections of the tuning set and of the several t^* hyperrectangles in any two-dimensional space defined by the binary combination of some of the x_i variables.

Through interaction with the user and a careful inspection/analysis of the above data summaries, it will hopefully be possible to identify one or more of the t^* zones in the feature space as specially promising from a practical implementation point of view. This selection process can be semi-automated by imposing additional constraints, determined by the decision-maker, and focusing his/her attention on the subset of t^* alternatives that satisfy them. These are some examples of the possible types of constraints that one may want to establish:

- Keep only those alternatives that have $H1(t^*) > 0.85$, which defines the minimum level of reliability necessary for a given hyperrectangle to deserve further consideration.
- Keep only those alternatives whose definitions involve ranges of values for some of the features above a given level. For instance, although a region associated with "maintaining the temperature at [290.0 K; 290.1 K]" may sound very appealing according to several criteria [for example, $H1(t^*)$], it will be useless if it is known *a priori* that control limitations make it infeasible to accomplish the goal of bounding the temperature within such a narrow bandwidth.

Those of the t^* alternatives that satisfy the additional constraints can then be grouped together according to the classes they are associated with, and within each group ordered by decreasing values of $H3(t^*)$. Instead of concentrating on a single "best" improvement opportunity, such as would be the case if only the surviving alternative associated with a desired class and with the highest $H3(t^*)$ score were to be kept, we strongly believe that it is better to interact with the decision-maker and consider all the above subsets of possibilities. This will offer him/her the chance of examining them, as well as being the responsibility for making a final choice among different possibilities and potential courses of action enumerated previously. This interaction is one of mutual benefits: from his own side, the user can get a better understanding of the process by examining the several uncovered improvement opportunities, even if only one of them is going to be followed; the input of invaluable specific domain expertise in the analysis of the results is extremely important. Without this validation and personal involvement, it is unlikely that any of the uncovered improvement suggestions will ever become actually implemented.

Extensions to the Basic Problem Formulation

In this work, we decided to focus on the discussion of our methodology on the premise that the quality improvement could be formulated as a classification task with an all-or-none degree of membership in each of the classes. This type of formulation is easier to visualize/explain, and thus it is particularly useful for a first exposure to the methodology. Also, there are some practical situations where it models the reality quite well (for example, in the manufacturing of components, one may consider that either the parts fit together or they don't, so that no intermediate situations need to be taken into account). Although the definition of crisp classes is still common in the quality arena, it is becoming increasingly clear that such an artificial division is not particularly meaningful. In the real world, concepts, rather than an all-or-none type are graded. The idea that any product is equally good within specifications and equally bad outside them needs to be revised. As pointed out by Deming (1986) and Taguchi (Roy, 1990; Phadke, 1989) among others, there is always some loss associated with deviations from a given target value, and the more one deviates from that target, the bigger becomes the associated loss. Thus, just being within specifications is not enough, and the concept of continuous improvement is associated strongly with this idea of trying to keep values closer and closer to the desired nominal targets. The utility of problem representations that reflect these facts has been claimed both within the quality management and fuzzy set communities. For instance, if the quality measurement variable, y (as is almost always the case in the processing industries) is real-valued, a crisp separation of its values into different classes leads to an unnecessary loss of information. A formulation of the same problem, where classes are identified as fuzzy sets or where one tries to learn directly the behavior of y (including both its level and dispersion) as a function of $\boldsymbol{x}$, will provide a better model of the reality in most cases. Extensions of our methodology to handle both of these types of problem formulation keep the basic structural backbone and follow the same fundamental steps as before. Here we will enumerate some of the main conceptual differences/similarities between the previous *classification* formulation of the quality improvement problem, and *fuzzy sets* and *function learning* formulations. The corresponding procedural adaptations will also be outlined (further details can be found in Saraiva, 1991).

Fuzzy sets

To reflect the graded nature of classes (a "high" y is more or less "high' depending on its particular value), we will associate with the y performance measure of any given example, $[(x_1, x_2, \ldots, x_M); y]$, the corresponding degrees of membership in all the K classes, $\mu_k(y)$, $k = 1, \ldots, K$, with $0 \le \mu_k(y) \le 1$. The induction of decision trees proceeds like as with crisp classes; but, in Eq. 2 the probabilities $P_k(t)$ are given by the ratio of the sum of degrees of membership for class k to the total sum

of degrees of membership for all the K classes. Both of these sums are evaluated over all the members of node t:

$$P_k(t)=\frac{\sum_{n=1}^{N(t)}\mu_k(y_n)}{\sum_{k=1}^{K}\left[\sum_{n=1}^{N(t)}\mu_k(y_n)\right]} \tag{9}$$

Nodes are considered as terminal whenever a particular class is predominant [for example, $P_k(t)>0.9$, for any single k] or contains a reduced number of examples. The leaves thus created act as probabilistic classifiers and are labeled with conditional probabilities $P(k/t)$ determined from the tuning set according to an expression equivalent to Eq. 9. Knowledge under this type of formulation will be expressed in the form of rules such as:

> "*If* $x_3<345$ *and* x_6 is decreasing *and* x_5 is true,
> *then* the corresponding y value may belong to
> *class 1*, with 0.7 probability
> *class 2*, with 0.2 probability
> *class 3*, with 0.1 probability."

During the pruning phase, Eq. 7 is employed. In the evaluation of $R(T_t)$, the number of misclassifications is replaced by the sum of the *absolute normalized errors* over all the examples assigned to leaves of T_t. For each of these examples, (x_i, y_i), assigned to leaf t, we designate as *absolute normalized error* half the sum over all the K classes of the absolute deviations between the normalized degrees of membership for case i, $\mu_{k,n}(y_i)$, and the corresponding $P_k(t)$ values (Saraiva, 1991).

Similar adaptations are considered for the definition of the active memory of exemplars and its on-line adaptation. Specifically:

- Instead of using only the closest neighbor to predict the normalized degrees of membership to be associated with a new incoming case, an interpolation procedure is employed. It takes into account a given number (for example, 5) of closest examples. The predictions are determined from the weighted averages of the degrees of membership associated with each of these closest examples. The weights given to the different examples in the above computation are negatively correlated with their distance to the new incoming pattern x.
- All the previous measures of accuracy, based on the determination of error rates, are now replaced by normalized prediction errors, of which the *absolute normalized error* presented above is a typical example. With this replacement, it becomes possible to follow the same steps as in the classification formulation to define the D metric, evaluate the performance of each exemplar, update the active memory, and determine the several evaluation scores associated with each of the improvement suggestions. For instance, the reliability of an exemplar is measured by the average normalized error observed when that particular exemplar happened to be the closest neighbor included in the interpolation procedure. Similarly, a new incoming case is added to the incremental memory of new exemplars whenever its prediction-normalized error exceeds a given threshold (for example, 0.3).
- To initialize the active memory of exemplars, the search for Tomek links is replaced by the application of a modified *condensed nearest neighbor* (Hart, 1968) procedure. Basically, examples are added iteratively to the active memory, until we find a subset of the initial training set, L_{tr}, which is informative enough for the interpolation process of estimation supported by it to provide absolute normalized errors inferior to some threshold (for example, 0.2) for all the remaining members of L_{tr}. This screening process provides automatically the active memory with a higher density of stored exemplars in the zones of the feature space where the mapping between degrees of membership and x is particularly complex.

Function learning

In a third alternative formulation, one can try to learn directly the behavior of y (including both its location and dispersion) as a function of x. Under a *function learning* formulation, the goal is still one of identifying significant hyperrectangles in the feature space, but this time each of them will be labeled directly with an approximation of the behavior of y that is expected to take place within that zone of the feature space. For example,

> "*If* $x_3<345$ *and* x_6 is decreasing *and* x_5 is true,
> *then* y will have an *average* of 245.5 and a *standard deviation* of 10.6."

In the construction of *regression decision trees* from a set of learning examples, at each node we try to find the split that reduces the y variability most. This points out the need to redefine $\phi(t)$. If we designate the average y value taken over all the learning examples contained in node t as $y_{avg}(t)$, then $\phi(t)$ can be expressed simply as:

$$\phi(t)=\sum_{i=1}^{N(t)}[y_i-y_{avg}(t)]^2 \tag{10}$$

The different leaves are labeled with both the average $[y_{avg}(t)]$ and standard deviation $[\sigma(t)]$ evaluated over the y values of the learning examples contained in t. A node is considered as terminal, if it contains a reduced number of examples *or* the standard deviation is considerably smaller than the average [for example, $\sigma(t)\leq 0.01\,y_{avg}(t)$].

After scaling all the y values, for example, to [0,1], most of the adaptations employed for fuzzy sets also hold within this function learning formulation. The *absolute normalized errors* are simply the absolute deviations between the real and estimated normalized y values. They are used, as before, to guide the pruning process, identify the distance measure D, evaluate the stored exemplars and update the active memory, and determine the scores associated with each of the final uncovered hyperrectangles.

Through an interpolation procedure similar to the one used for fuzzy sets, the y values associated with new incoming vectors x are estimated as the weighted average of the corresponding y values for a number (5 to 10) of closest neighbors. Also, a process analogous to the one described for fuzzy sets is employed to build an initial active memory where the density of points increases with the local complexity of the mapping between x and y.

Taguchi (1989) has pointed out the need for taking also into account the dispersion of y, rather than only its level, to identify operating strategies that are robust to environmental variability and insensitive to uncontrollable manufacturing factors. He

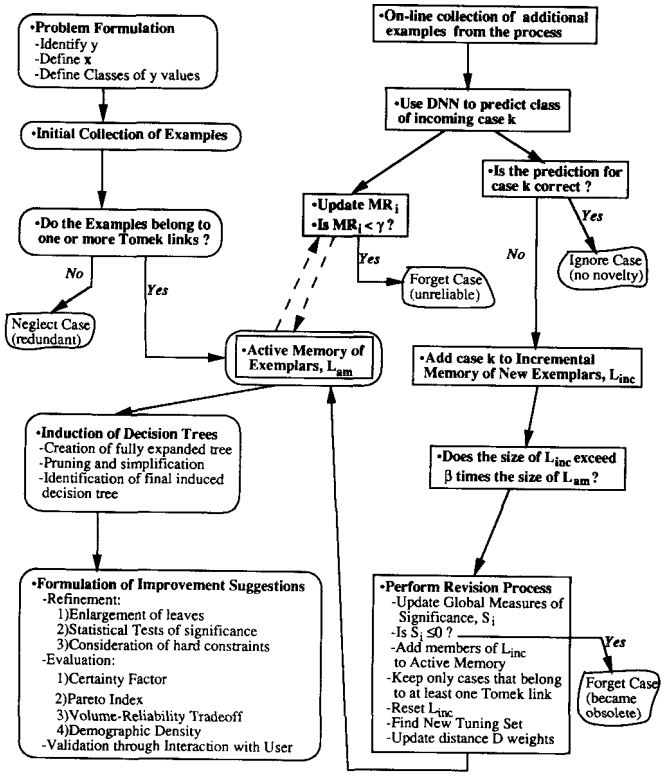


Figure 7. Comprehensive functional diagram of the methodology.

also suggested the use of a *signal to noise ratio* as an index that lumps together both accuracy and precision issues. Thus, in handling function learning formulations, we will consider an additional evaluation score for each improvement alternative, the *Taguchi index, H5(t*)*. It is simply the signal to noise ratio (Roy, 1990) associated with the t^* hyperrectangle:

$$H5(t^*) = -10\log_{10} MSD(t^*) \tag{11}$$

where $MSD(t^*)$ stands for the mean squared deviation from the target y value, taken over all the examples from the tuning set that lie inside t^*. Other definitions of signal to noise ratios for y variables such as yield and energy consumption, needed to achieve the highest/lowest possible values rather than a given prespecified target, are given by Saraiva (1991).

Results of Case Studies

In this section we will present four different case studies to illustrate several components of the methodology, starting with the overall picture of our learning framework. In the previous sections, different steps taken were detailed. Now, it is possible to go one abstraction level down and provide a comprehensive functional overview with a higher resolution than the original Figure 1. Such a summary diagram is shown in Figure 7. Since all the case studies that we will discuss fall under the category of pattern recognition problems with crisp class definitions, the diagram also corresponds to this type of approach. Minor changes would have to be introduced either for fuzzy sets or function learning formulations to reflect the adaptations discussed previously. Figure 7 also shows the interactions between the inductive generalizations and analogical reasoning components. More detailed descriptions of the internals for each block can be found in earlier sections on the Induction of Decision Trees, the Adaptive and Case-Based features of the methodology, and the Formulation of Improvement Suggestions from Induced Decision Trees.

For easier understanding and visualization, in all of the following case studies we will focus on crisp class definitions and problems where a projection of the data in a carefully chosen two-dimensional feature space provides the possibility of examining the problem structure and present/evaluate the results of our approach. Next, we will discuss the studies performed on the following systems:

(a) As the first very simple and tutorial example, we will consider a simulated CSTR where operating data were generated by a Monte Carlo procedure. This case will be analyzed from several perspectives to illustrate most of the concepts previously presented.

(b) Industrial operating data collected from a refinery unit are used as the example of a specific situation where by exploring existing information it was possible to formulate promising improvement suggestions. We will also show how the extracted knowledge might guide the execution of a reduced set of confirmatory designed experiments. The steps of refinement and evaluation reported earlier will be examined with particular detail in this case study.

(c) Records of values obtained from the continuous kraft digester of a pulp mill will also be analyzed. They are presented here to emphasize the importance of the significance testing tools and validation steps, because they prevent the formulation of unreliable generalizations and the creation of fictitious knowledge from data that do not convey enough information to support it.

(d) Finally, we will employ the same CSTR as in (a) to simulate a permanent on-line implementation of the learning methodology, study in detail the behavior of its analogical reasoning component, and the corresponding evolution of memory contents under a number of different operation contexts.

Simulated CSTR

The system considered here is a simple CSTR, where a first-order irreversible reaction following an Arrhenius law is assumed to occur, converting reactant A into product B. We will define as our quality measure ($[Bf]$ or y) the concentration of product B in the output stream. It was further assumed that no B is present in the input stream and that the reaction follows a stoichiometry $A \rightarrow B$.

Four operating variables, on which the value of y depends, are involved: (a) the temperature in the reactor (T or $x1$); (b) the concentration of reactant A in the input stream ($[Ai]$ or $x2$); (c) the level of liquid in the reactor (L or $x3$); and (d) the volumetric flow of both the input and output streams (Q or $x4$). To generate operating data, we took these four features as random variables having Gaussian distributions with the following means and standard deviations: $T \sim N(300\ K, 0.5\ K)$; $[Ai] \sim N(0.8\ \text{mol/dm}^3, 0.05\ \text{mol/dm}^3)$; $L \sim N(13\ \text{m}, 0.5\ \text{m})$; $Q \sim N(10\ \text{m}^3/\text{s}, 0.5\ \text{m}^3/\text{s})$. Using this Monte Carlo simulator, 800 examples (x_i, y_i) were created, 400 of them being assigned to a *training set*, L_{tr}, and 200 to each of both a *tuning*

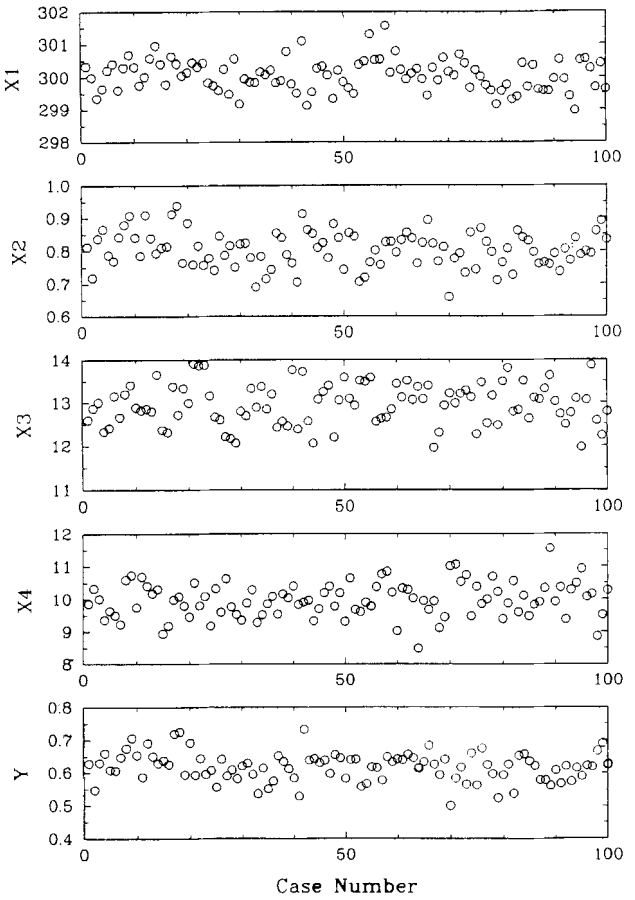


Figure 8. Values corresponding to the first 100 examples of the training set.

set L_{tu}, and an independent *testing set,* L_{te}. In Figure 8, we have plotted the first 100 members of L_{tr}. It is clear, due to the problem definition as well as by looking at the data time series, that the process is under statistical control. Following the common *statistical process control* paradigm, this is the kind of situation where one would conclude that the system is already performing at its best level and believe that no further improvement could be achieved. It must also be recognized that just by visual inspection of the data it is virtually impossible to identify significant common causes and formulate improvement suggestions in terms of any combination of the operating variables, $x1$ to $x4$.

Three different classes of y values were defined, corresponding respectively to "low," "normal" and "high" output concentrations of the reaction product B. It is assumed that the y values are already centered around the target value, and thus we considered as "normal" all those y values that did not deviate more than one standard deviation from either side of the average. Following this criterion, class 1 contains all instances with $y_i \leq 0.575$, class 2 includes all points with $0.575 < y_i < 0.656$, and all examples with $y_i \geq 0.656$ are considered to belong to class 3. In Figure 9a, we present a projection of the 400 training points in the $x1$–$x2$ coordinates, with identification of the corresponding classes. Although these two variables are not enough to produce a complete discrimination, there are clearly zones of the $x1$–$x2$ plane where it is very likely to get points from only one particular class. Also by visual

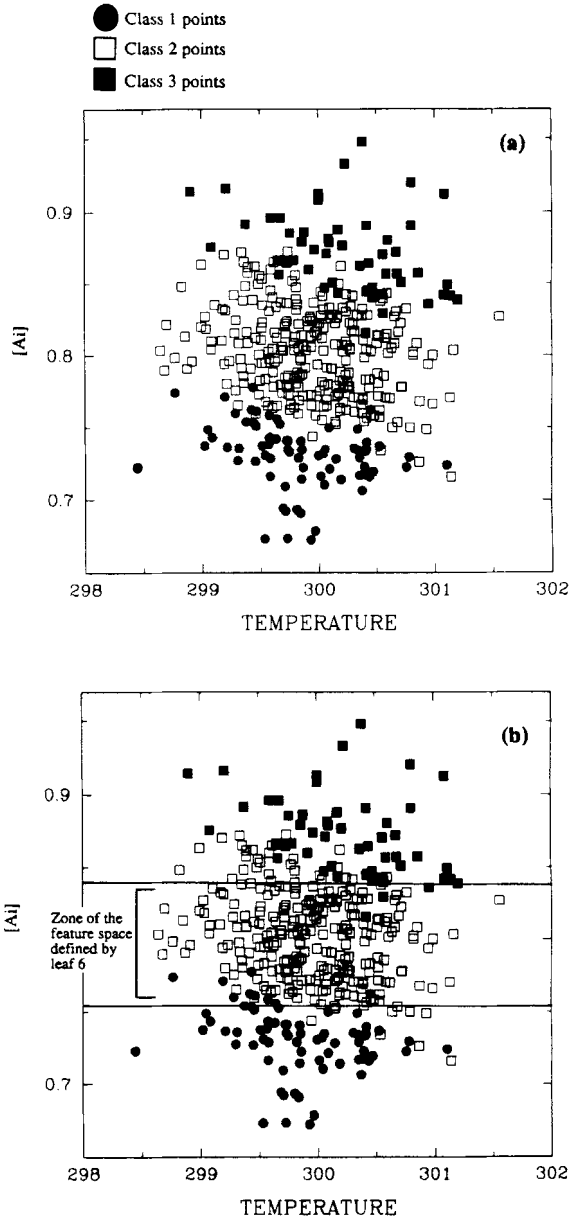


Figure 9. Simulated CSTR example: (a) projection of the training set on the $x1 - x2$ plane and (b) improvement suggestion associated with leaf 6.

inspection one can intuitively formulate an immediate improvement suggestion: reducing the variability of $x2$ and keeping the values of this feature in the range [0.75, 0.85] should lead to almost only "good" points. It is exactly this type of knowledge that we hope our methodology might be able to capture automatically, even in multidimensional spaces where no visual inspection can be performed.

All the 400 members of the training set were used to create a fully expanded tree, T_{max}, and then to build the corresponding sequence of pruned decision trees, $\{T_i\}$. According to the accuracy levels achieved by these pruned decision trees, when used to classify the cases contained in the tuning set, a *final pruned decision tree,* T^f, was found. It is shown in Figure 10. From T^f we can extract directly supervisory control rules and suggestions for process improvement. For instance, doing a

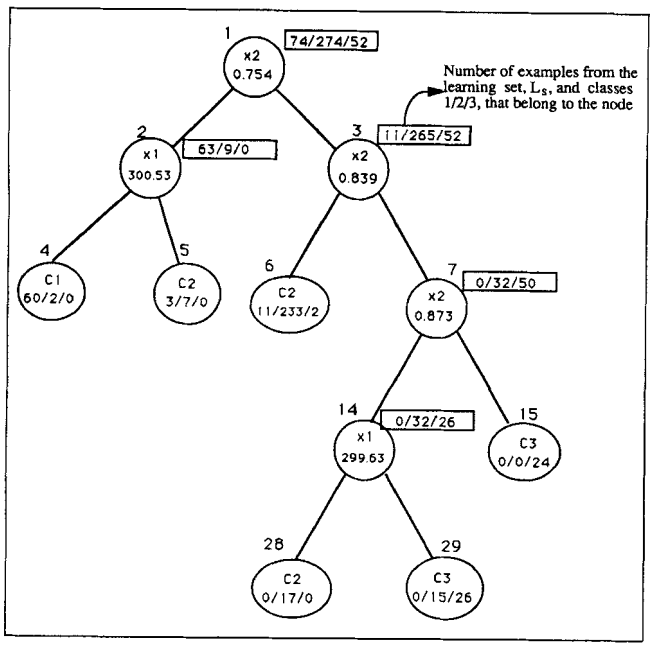


Figure 10. Final induced classification decision tree.

search over the class 2 leaves, the evaluation scores associated with node 6 are found to be particularly high: $H1(6) = 0.95$, which means that within this constrained zone of the feature space around 95% of the production will belong to class 2, as opposed to only 68% under the present operating conditions; $H2(6) = 0.85$, which implies that about 85% of the currently obtained good points are generated within this same zone of

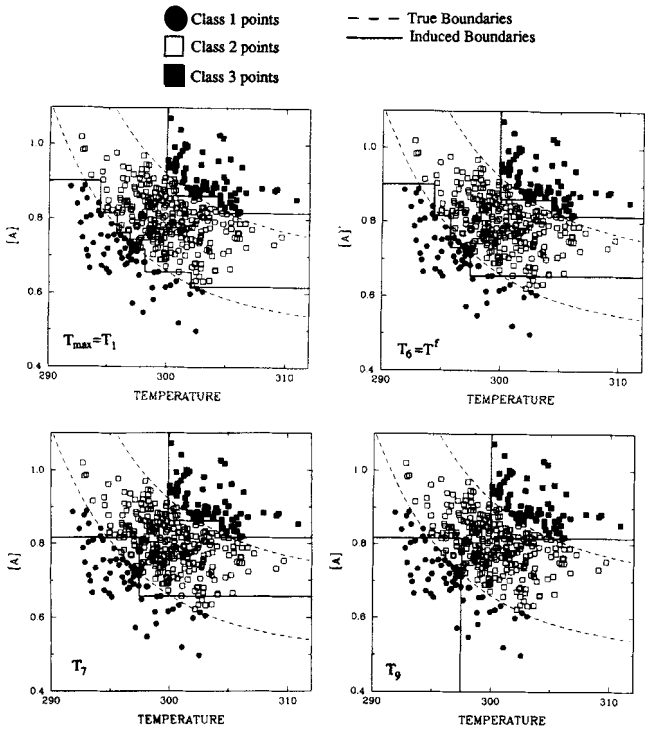


Figure 11. Descriptions of the feature space at different levels of detail.

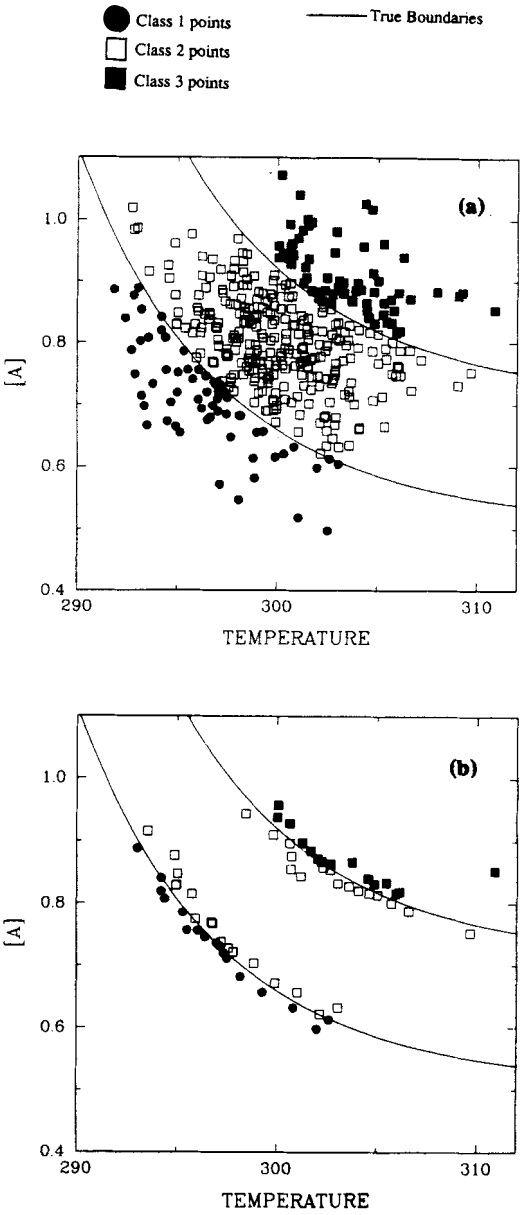


Figure 12. (a) Complete set of training examples and (b) location of the exemplars stored in the active memory.

the space. The improvement suggestion associated with this particular leaf is extremely similar to the one formulated earlier by visual inspection, the primary goal thus being to keep $x2$ within [0.754, 0.839] (Figure 9b). All the subsequent refinement and validation steps preserve this rule essentially unchanged. Enough evidence seems to be present in this case to lead directly to its implementation through the introduction of process changes, if indeed it is feasible to reduce the $x2$ spread.

To visualize and compare the induced boundary decision lines with the true ones, we repeated the previous study, but this time keeping both $x3$ and $x4$ constant and equal to their nominal values, and considering $T \sim N(300\ K,\ 3.5\ K)$, $[A] \sim N(0.8\ \text{mol/dm}^3,\ 0.1\ \text{mol/dm}^3)$. Three classes of y values were defined following the same criterion as before. We provide in Figure 11 a sample of the different levels of detail that

are associated with the partitions of the feature space performed by the different members of the $\{T_i\}$ sequence.

Moving now to the analogical reasoning component of the methodology, we found the set of optimal D weights to be given by $w_1 = 0.75$ and $w_2 = 0.82$. Using this distance metric, only 65 exemplars (out of the initial total of 400 cases contained in the training set, L_{tr}, shown in Figure 12a) were identified as deserving to be stored in the active memory of exemplars, L_{am}. In Figure 12b, we represent those 65 exemplars, which are clearly points that convey crucial information for the definition of decision boundaries. Using only these exemplars to induce decision trees does not affect the quality of induced generalizations. We determined and examined the decision boundaries identified by the final T^f trees, obtained respectively when (a) all the 400 examples contained in L_{tr}, and (b) only those 65 exemplars stored in L_{am} were used to construct these decision trees. They are extremely close to each other and even difficult to differentiate visually. Both induced decision trees are also very similar and achieve a common error rate of 0.08 when applied to classify the members of the testing set, L_{te}.

Refinery unit

As the second application example, we will consider records of operating data collected from a refinery unit (Daniel and Wood, 1980). The y variable is the octane number of the gasoline product, and the corresponding four operating features are three different measures of the feed composition ($x1$, $x2$, $x3$), and the value of an unspecified operating condition ($x4$). A total of 79 examples, collected under a "normal" operating context, were available. From these, 47 randomly chosen cases were used as the learning set and active memory of exemplars, L_{am}, from which generalizations were induced. The remaining ones were included in a separate tuning set, L_{tu}. In this problem, one wishes to achieve values of y as high as possible, and three different classes were considered as follows: class 1 ("very low") for $y \leq 91$; class 2 ("low") when $91 < y < 92$; and class 3 ("good") for $y \geq 92$. In Figure 13a, we present the final induced decision tree, T^f, obtained after the pruning phase. Also, in Figure 13b, we show the partition of the feature space associated with the leaves of T^f, together with the projection of all the available 79 examples on the $x1$ vs. $x4$ plane. It seems that these two operating variables determine the current performance of the refinery unit, and T^f indeed performs a reasonable division of the plane. To get a better performance, we must search for operating zones that will result in mostly class 3 outputs (high-octane gasoline). Terminal nodes 2 and 7 identify two of such zones and uncover promising improvement opportunities, which we will designate as (1) and (2), respectively:

(1) Keep $x1$ in the lower tail of its current distribution: $x1 \in [44.6, 55.4)$.

(2) Keep $x1$ and $x4$ simultaneously in the higher tail of their current distributions: $x1 \in [55.4, 75.5]$ and $x4 \in [1.8, 2.3]$.

The constraint on $x1$ associated with the second suggestion is removed after going through the refinement expansion process described previously. We then obtain, as final alternatives: (1) $x1 \in [44.6, 55.4]$ and (2) $x4 \in [1.8, 2.3]$. Both of them survive the additional Fisher's exact test of significance. Due to the small number of examples available, no screening procedure was followed to identify the active memory of exemplars. Thus,

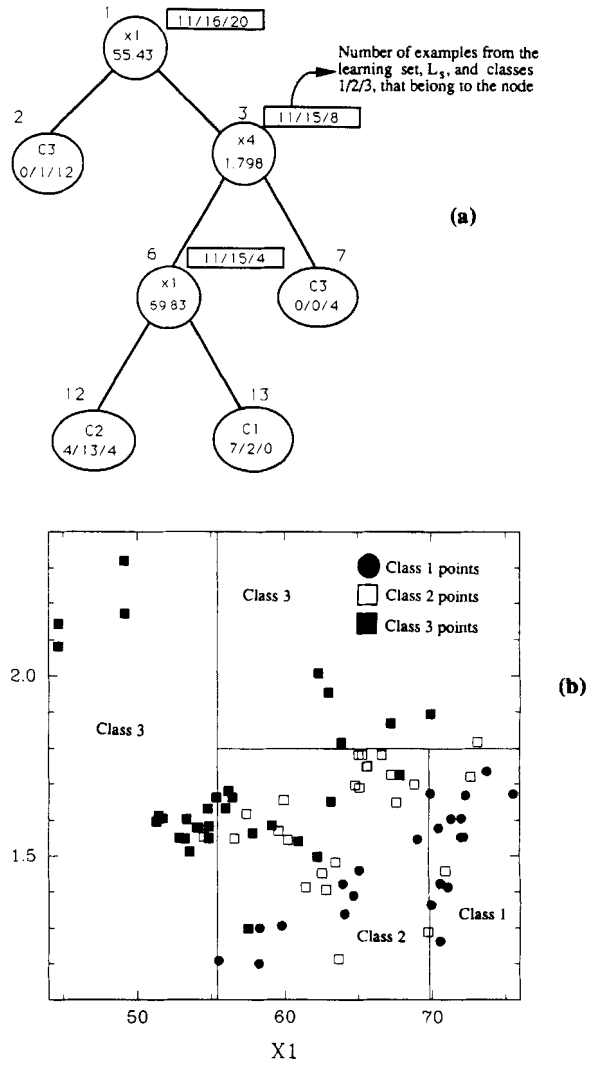


Figure 13. (a) Final induced classification decision tree for the octane example and (b) partition of the feature space associated with leaves of the tree.

the learning set is just a random sample taken from the process, and the different evaluation scores can be determined directly from it, leading to the following results:

Suggestion (1)	*Suggestion (2)*
$H1 = 0.9$	$H1 = 1.0$
$H2 = 0.60$	$H2 = 0.35$
$V = 0.35$	$V = 0.45$
$H3 = 0.97$ (for $\tau = 0.2$)	$H3 = 1.09$ (for $\tau = 0.2$)
$H4 = 37.14$	$H4 = 15.56$

These values confirm the capabilities of any of the above suggestions as performance improvement candidates, and no clear superiority can be pointed out in favor of one or the other. The $H1$ scores tell us that we will get with almost full certainty only "good" y values, while operating inside these zones of the feature space. Also, most of the currently obtained class 3 values are associated with operating conditions contained inside the corresponding hyperrectangles ($H2$), which

have considerably high volumes (V) and demographic densities ($H4$).

As both alternatives seem equally appealing up to this point, a final implementation decision would require the input of specific domain knowledge, and considerations around (a) the feasibility and eventual difficulties associated with the introduction of the required changes in the process, (b) how these changes would affect the system performance according to other objectives (such as steam consumption, safety, and operating costs).

Alternatively, before any permanent implementation takes place, it can be decided to conduct a set of confirmatory experiments. For example, one might follow a 2^2 full factorial design, keeping unchanged the current policies for the remaining operating variables and considering $x1$ and $x4$ as the two factors to be studied, each of them having as levels: 50 (approximate midpoint for the range of values associated with suggestion 1) and 62 (approximate current operating average) for variable $x1$; 1.6 (approximate current operating average) and 2.05 (approximate midpoint for the range of values associated with suggestion 2) for variable $x4$. An analysis of the results of these experiments (considering as response directly the y values or the fractions of class 3 cases obtained at each of the corners of the square) will hopefully confirm the previous expectations and lead to the introduction of the appropriate changes in the process.

In conclusion, we deliberately excluded from our analysis three additional records of data (Daniel and Wood, 1980) that were collected while a specific feed preparation unit was shut down for repairs. During this period, the process went out of statistical control, feedstock with different compositions (very low $x1$ values) was temporarily processed, and this resulted in the production of gasoline with an unusual high octane number. This observation resulted in the introduction of process changes to bring the "normal" feed composition closer to the ones obtained during the shutdown. This corresponds essentially to an implementation of suggestion 1. The results previously described show that our methodology would have uncovered this improvement opportunity by analyzing only "normal" operating data, presumably long before and even if no occasional shutdown had taken place.

Continuous pulp digester

In the third example, we also explored real industrial data, which illustrate two main points: it is crucial to use the refinement, evaluation and validation steps of the methodology to prevent the formulation of fictitious or unreliable knowledge; adaptive and on-line capabilities are important to keep searching for improvement opportunities as new data become available.

The results were obtained from the treatment of operating data collected during two weeks at the continuous kraft pulping production facilities of SOPORCEL, a Portuguese pulp mill. The installation contains both a separate *impregnation vessel* and a *vapor-phase digester*. Only situations of a constant rate of production were considered, and the *kappa number* (an index determined off-line on the laboratories from samples of product) of the pulp was taken as our y variable, the objective being to have a reduced kappa variability and keep its value as close as possible to a desired target. Several studies were performed, considering different groups of operating variables. For brevity, we will focus on the simpler problem definition, where the following two operating features were considered: alkali charge (ratio between the amount of chemical reactants added and wood transformed) designated as CA or $x2$; H-factor or $x1$ (a semi-empirical quantity that combines in a single parameter the residence times and corresponding temperature profiles experienced by the chips across the impregnator and the digester). As a particular chip takes about 4 hours to go through all the equipment, in the definition of each example the values of y and $x1$ were combined with the corresponding $x2$ measured 4 hours earlier.

In Figures 14a and 14b, we present the scatterplots of y vs. each of the operating variables. No clear mutual dependency relations can be found; accordingly, the corresponding correlation coefficients are very low. Different linear and nonlinear regression models were also fitted to the data, but failed to provide reliable predictions and resulted in almost zero coefficients of determination. All these preliminary observations point out the lack of structure in the data. Given the tight control over the operating variables, it appears that measurement errors and/or other unknown factors are responsible for most of the observed y variability. This is the sort of situation that one would hope our methodology would automatically associate with the third of the courses of action enumerated earlier (simply keep incorporating new data until reliable and safe improvement opportunities can be found).

To follow the pattern recognition approach to process improvement, three classes of y values were defined: class 1 ("low" kappa) for $y \leq 15.0$; class 2 ("good" kappa) for $15.0 < y < 15.7$; and class 3 ("high" kappa) for $y \geq 15.7$. In Figure 14c, we show the distribution of the learning set (with indication of the corresponding classes) in the feature space. Confirming our earlier expectations, the operating variables considered are not able to discriminate the different classes, and it is not possible to identify with enough reliability zones of the feature space that one would associate mostly with any particular class of y values. However, it is always possible to build a fully expanded decision tree, T_{max}, from data such as the above, by iteratively splitting the feature space until an overfitted partition is found where points from only a single class are contained in each terminal node. For instance, in Figure 14d, we present the first layers of the decision tree T_{max} induced from the set of 68 examples in Figure 14c. The full tree includes more than 20 leaves, each of which containing five examples at most. Any inductive tool can become very "dangerous" and generate spurious generalizations, unless it is used with caution. Thus, as important as the ability to formulate useful improvement suggestions, is the existence of validation mechanisms that prevent one from uncovering unreliable and unsafe generalizations, or creating fictitious knowledge where none is present. Due to the lack of structure in the data when a separate tuning set of examples is considered, none of the leaves of T_{max} survives the first of our testing procedures, the pruning process described earlier. The final induced decision tree, T^f, was thus found to be a degenerate one, consisting only of the root node where all the learning examples are included. As we expected, the message that our methodology would convey to the operator is simply that no reliable improvement suggestions could be formulated from the records collected during those two weeks due to the lack of structure found in the data. It

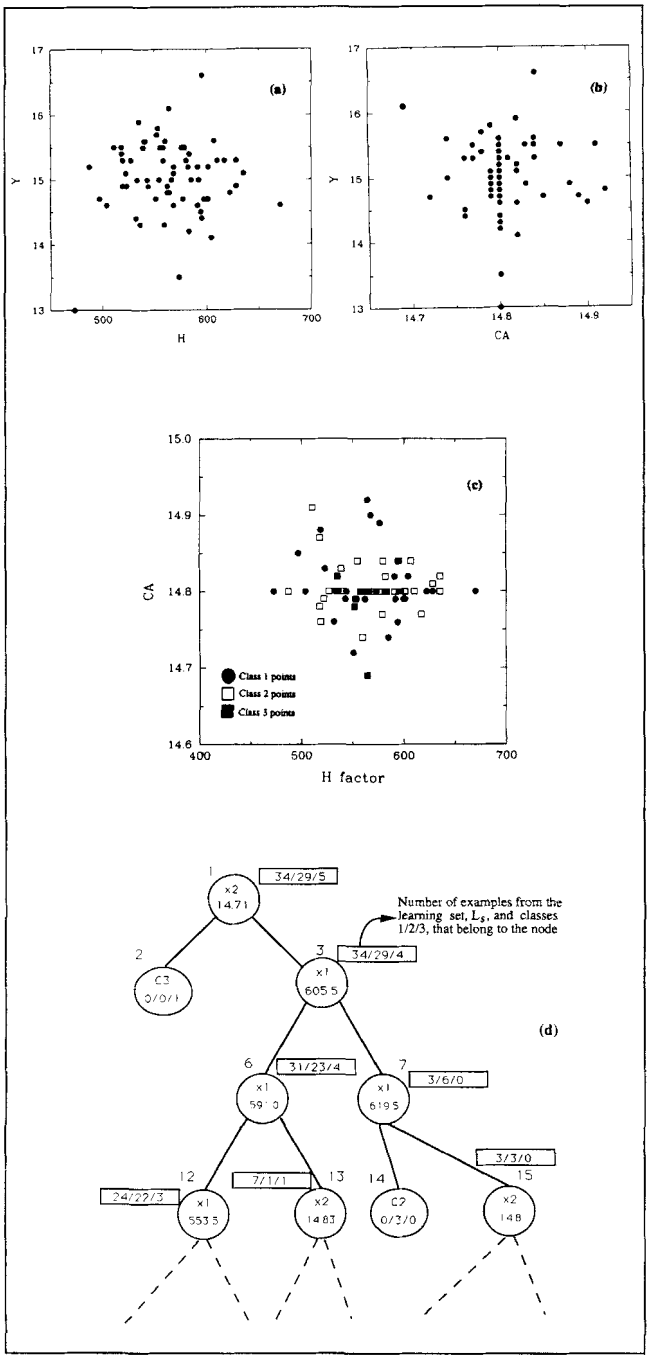


Figure 14. Characteristics of the pulp digester example: (a) scatterplot of *Y* vs. *H*, (b) scatterplot of *Y* vs. *CA*, (c) location of examples in the feature space, and (d) first layers of fully expanded classification decision tree.

might then proceed by exploring new incoming examples, until such improvement opportunities can be detected.

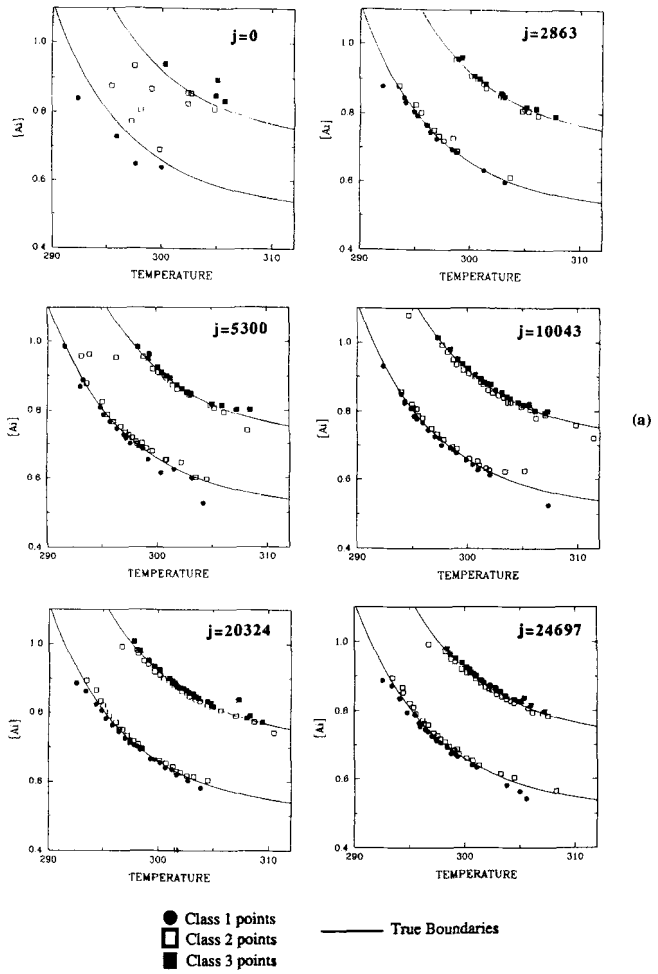
Finally, we would like to stress that even if any of the leaves happened to survive the pruning process, they would still have to go through all the other additional testing and evaluation procedures enumerated earlier. Only clearly reliable, safe and promising alternatives will be presented to the decision-maker.

Memory evolution in a simulated CSTR; on-line implementation

To analyze the behavior of the adaptive process described earlier and examine the evolution of the dynamic memory contents, we used the previous CSTR model to simulate the on-line implementation of our learning methodology. To visualize the active memory of exemplars and its evolution, as well as the true boundary decision lines, a two-dimensional feature space was considered. Thus, both $x3$ and $x4$ were held constant and equal to their nominal values ($L = 13$ m and $Q = 10$ m^3/s), while for the remaining operation variables we took $T \sim N(300\ K,\ 3.5\ K)$, and $[Ai] \sim N(0.8$ mol/dm^3, 0.1 mol/dm$^3)$. In all the trials conducted, three different classes of y values were defined, according to the previous standard deviation criterion. We will designate as j the number of a particular case, generated j sampling times after the learning procedure is supposed to have been implemented on-line in the process. With respect to the parameters necessary to define the adaptation process fully, we took $\beta = 0.35$, $\gamma = 0.75$, and a time horizon of 500 cases. Each tuning set consists of a total of 200 examples.

In our first run, the decision boundaries were kept fixed, and the reactor operation was simulated through the generation of a sequence of 30,000 cases. To initialize the memory, a set of 30 warming-up examples was taken into account. This resulted in a starting memory (for $j = 0$) that contains a total of 18 exemplars, shown in Figure 15a. The dynamic memory evolution was examined, and some snapshots of its contents, after the conclusion of different revision processes, are presented in Figure 15a. As time goes by and more raw experience is accumulated, the stored exemplars become placed closer and closer to the true decision boundaries. At around $j = 5{,}000$, the members of the active memory already provide a reasonable support for the identification of decision boundaries that reproduce accurately the true ones. As can be concluded from Figure 15b, which shows the active memory cardinality vs. time, there seems to be an exponential growth phase, followed by an almost stationary phase, reached at about $j = 10{,}000$, and centered around a total of 90 exemplars. Although we are dealing with relatively simple true decision boundaries and just a two-dimensional feature space, it is still remarkable that a total of only about 90 carefully selected exemplars seem to condense and convey most of the useful domain information contained in more than 30,000 data points. As evidenced by Figure 15c, the error rate obtained when DNN predicts the membership of the cases contained in the tuning sets, after each revision process is concluded, decreases rapidly until about $j = 5{,}000$, and from there on oscillates around only 0.02.

As a second trial, aimed at observing the memory dynamics when a modification of the underlying physical process takes place, we repeated the previous experiment, but this time emulating a sudden change of regime. More specifically, we allowed the reactor to run with an activation energy of 99.77 kJ/mol, as before, for $j < 10{,}000$. Then, the activation energy was moved instantaneously to its new value, 101.43 kJ/mol and kept fixed at it for the rest of the exercise. During the whole process, the y values used to identify the different classes were kept constant and equal to the ones previously employed in the first experiment. Consequently, the shift of activation energy is accompanied by a movement of the true decision boundaries across the feature space. In Figure 16a, we describe



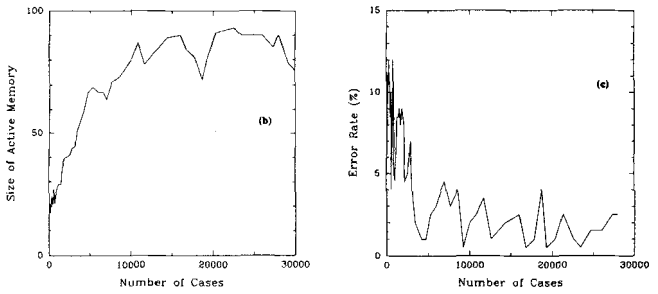
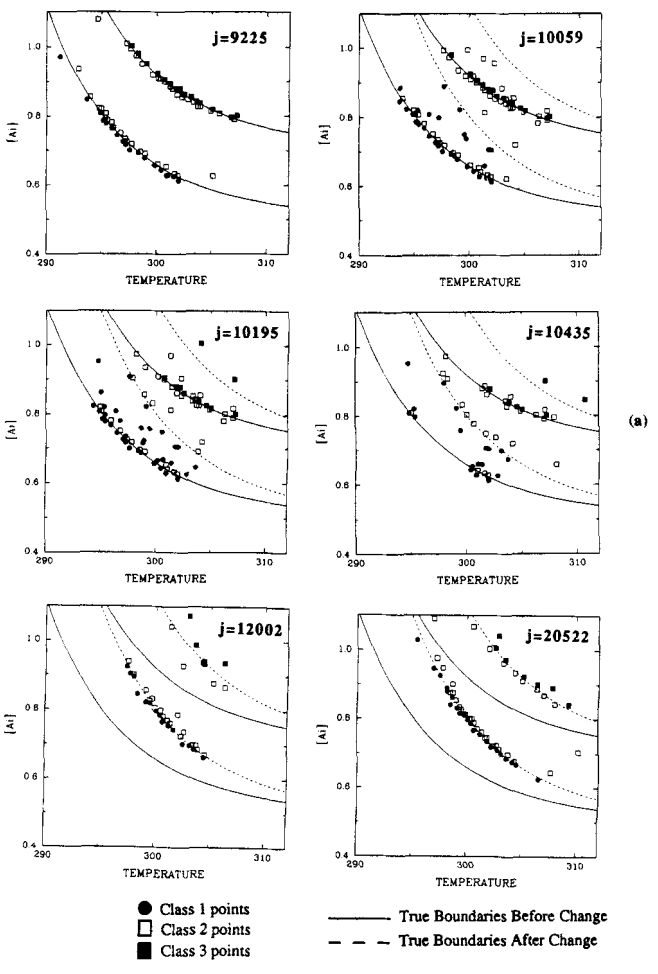


Figure 15. Dynamic evolution of (a) memory contents for fixed boundaries, (b) active memory size, and (c) DNN error rate, for the simulated CSTR.



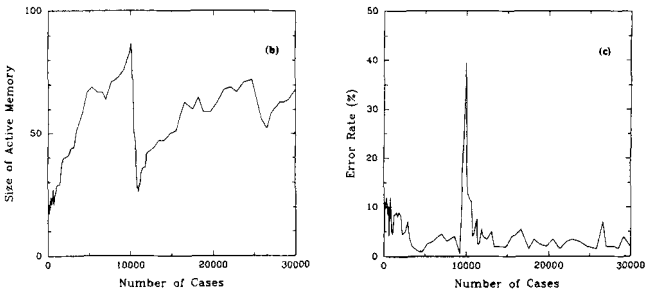


Figure 16. Physical process changes and their effect on the dynamic evolution of (a) memory contents around $j = 10{,}000$, (b) active memory size, and (c) DNN error rate, for the simulated CSTR.

the active memory contents immediately before and after the drastic modification took place. It is interesting to note how examples are gradually accumulated around the new boundaries and replace their obsolete counterparts. A peculiar phenomenon that we decided to designate as "*social security effect*" is particularly visible at the memory snapshot taken for $j = 10{,}435$. To understand this effect, let's focus on a few cases of class 2 that lie close to the old 1-2 decision boundary. These are exemplars that used to provide an important and valid contribution to the active memory, before the process modification took place. Thus, at $j = 10{,}000$, they possessed quite high measures of reliability and had already been used to predict the membership of a large number of incoming cases. It would take a very long time, after $j = 10{,}000$, for the MR_i scores to decrease significantly and force their withdrawal from memory. Neither did enough time elapse for the S_i scores to become nonpositive, but it is already clear at $j = 10{,}435$ that something has changed. The feature space zone around these "veteran" points seems to belong now to class 1. To prevent further prediction errors to occur, groups of younger class 1 examples immediately surround and protect the older exem-

plars that have become obsolete. Once more than a time horizon period of time (in our studies, 500 cases) goes by, after the change took place, a new revision point is reached; the S_i scores of the "veterans" become nonpositive, and they are finally forgotten. In Figures 16b and 16c, we present the plots of memory size and DNN error rate vs. time, respectively. At the critical time instant, $j = 10{,}000$, the memory size registers a sudden decline, while the error rate goes through an extremely high peak of about 0.4. As mentioned before, this type of behavior can be explored as a powerful diagnostic tool for the almost instantaneous detection of the occurrence of significant changes in the process. The adaptive algorithm reacts immediately to this revolution and manages to come back within a few revision processes to both the typical range of error rates for static situations and the usual size growth dynamics.

Two other studies were also performed with the goal of examining how the memory contents react to a gradual and evolutionary change of activation energy and the introduction of a sequence of examples contaminated with noise. They confirm both the adaptive capabilities of our analogical reasoning algorithm and its ability to rapidly identify and exclude noisy instances from the active memory. A detailed description of these additional runs may be found elsewhere (Saraiva, 1992).

Conclusions

We have described a general and flexible methodology for interpreting and analyzing operating data, and searching for continuous process improvement opportunities. It can handle various types of variables and support problem formulations ranging from traditional crisp class definitions to function learning. It is deliberately a passive approach to existing records of data that do not require the execution of any sort of field experiments. Very few *a priori* and arbitrary assumptions or design decisions have to be made. Only the values of some parameters with clear and common-sense physical interpretation need to be chosen.

Although competitive with alternative techniques for the same tasks in terms of accuracy, the main advantages of this approach are related to representational issues: the output-induced knowledge is presented explicitly, using an operational and symbolic description language. This leads to the immediate and transparent identification of process improvement opportunities.

The overall methodology integrates symbolic inductive learning and case-based reasoning components, and combines intrinsic advantages of both of these schools of approach to machine learning. Analogical reasoning is used to guide the inductive process, by keeping an active memory of exemplars, updating it as new incoming data become available, identifying novelty, removing noisy or unreliable instances, and determining when to revise the current induced generalizations. Decision trees are constructed from the examples contained in the active memory. The knowledge captured by them is then converted into a set of improvement suggestions that are successively refined, evaluated, and validated. Finally, they can lead either to the introduction of changes into the process or the execution of a reduced set of confirmatory experiments, preceding any permanent implementation.

Results obtained from both simulated and real industrial operating data were used to illustrate the concepts presented and the potential practical capabilities of the methodology.

Further refinements and extensions are currently under development and will be described in future publications. These will include the simultaneous consideration of several operating objectives for a single system, as well as addressing the interactions and propagation of the learning process through various systems.

Acknowledgment

Financial and other types of support from the Leaders for Manufacturing program at MIT, Rotary Foundation, Fulbright Program, FLAD, and INVOTAN are gratefully acknowledged. We also deeply appreciate the availability of SOPORCEL that provided industrial data from its plant.

Literature Cited

Anderson, J., *Cognitive Psychology and Its Implications*, 3rd ed., W. H. Freeman, ed. (1990).

Box, G., and N. Draper, *Evolutionary Operation: A Statistical Method for Process Improvement*, Wiley (1969).

Breiman, L., et al., *Classification and Regression Trees*, Wadsworth (1984).

Daniel, C., and F. Wood, *Fitting Equations to Data*, 2nd ed., Wiley (1980).

Deming, W., *Out of the Crisis*, MIT, Center for Advanced Engineering Study (1986).

Fu, K., *Sequential Methods in Pattern Recognition and Machine Learning*, Academic Press (1968).

Garcia, C., and D. Prett, "Advances in Industrial Model-Predictive Control," *Chemical Process Control, CPC-III*, Morari and McAvoy, eds., CACHE-Elsevier (1986).

Glymour, C., *Discovering Causal Structure*, Academic Press (1987).

Goodman, R., and P. Smyth, "Decision Tree Design Using Information Theory," *Knowledge Acquisition*, **2**, 1 (1990).

Hahn, G., and A. Dershowitz, "EVOP Today—Some Survey Results and Observations," *Appl. Statistics*, **23**(2), 214 (1974).

Hart, M., and R. Hart, *Quantitative Methods for Quality and Productivity Improvement*, Quality Press (1989).

Hart, P., "The Condensed Nearest Neighbor Rule," *IEEE Trans. on Info. Theory*, 515 (1968).

Hayes, J., et al., *Machine Intelligence*, Vol. 11, Oxford Univ. Press (1985).

Hunt, E., *Concept Learning: an Information Processing Problem*, Wiley (1962).

Hunt, E., et al., *Experiments in Induction*, Academic Press (1966).

Hunter, J., "Statistical Quality Technology," *Philosophical Trans. of the Royal Soc. of London*, Ser. A, **327**, Special Issue on "Industrial Quality and Productivity with Statistical Methods," 599 (1989).

Juran, J., *Managerial Breakthrough*, McGraw-Hill (1964).

Kearns, M., *The Computational Complexity of Machine Learning*, MIT Press (1990).

Lasdon, L., and T. Baker, "The Integration of Planning, Scheduling and Process Control," *Chemical Process Control, CPC-III*, Morari and McAvoy, eds., CACHE-Elsevier (1986).

LaVigna, A., "Nonparametric Classification Using Learning Vector Quantization," PhD Thesis, Univ. of Maryland (1989).

Moret, B., "Decision Trees and Diagrams," *ACM Computing Surveys*, **14**(4), 593 (1982).

Muggleton, S., *Inductive Acquisition of Expert Knowledge*, Addison-Wesley (1990).

Nii, H., "The Blackboard Model of Problem Solving," *AI Mag.*, **7**(2), 38 (1986).

Park, Y., and J. Sklansky, "Automated Design of Linear Tree Classifiers," *Pattern Recognition*, **23**(12), 1393 (1990).

Phadke, M., *Quality Engineering Using Robust Design*, Prentice Hall (1989).

Quinlan, J., "Induction of Decision Trees," *Machine Learning*, **1**, 81 (1986).
Quinlan, J., "Simplifying Decision Trees," *Int. J. of Man-Machine Studies*, **27**, 221 (1987).
Riesbeck, C., and R. Schank, *Inside Case-Based Reasoning*, Lawrence Erlbaum (1989).
Roy, R., *A Primer on the Taguchi Method*, Van Nostrand Reinhold (1990).
Ryan, T., *Statistical Methods for Quality Improvement*, Wiley (1989).
Salzberg, S., *Learning with Nested Generalized Exemplars*, Kluwer Academic Publishers (1990).
Samet, H., "Hierarchical Representations of Collections of Small Rectangles," *ACM Computing Surveys*, **20**(4), 271 (1988).
Saraiva, P., "Computer-Aided Process Operation Analysis and Improvement," Thesis Proposal, Dept. of Chemical Engineering, MIT (1991).
Saraiva, P., "An Adaptive and Dynamic Memory of Exemplars," Laboratory for Intelligent Systems in Process Engineering, Internal Report, Dept. of Chemical Engineering, MIT (1992).
Schmidt, S., and R. Launsby, *Understanding Industrial Designed Experiment*, 2nd ed., AIR Academy Press (1989).
Shank, R., *The Cognitive Computer*, Addison-Wesley (1984).
Shank, R., *The Creative Attitude*, MacMillan (1988).
Shannon, C., and W. Weaver, *The Mathematical Theory of Communication*, 11th Printing, Univ. of Illinois Press (1964).
Shrager, J., and P. Langley, eds., *Computational Models of Scientific Discovery and Theory Formation*, Morgan Kaufmann (1990).
Taguchi, G., *Introduction to Quality Engineering*, Asian Productivity Organization (1989).
Taylor, W., *What Every Engineer Should Know About Artificial Intelligence*, MIT Press (1989).
Tomek, I., "Two Modifications of CNN," *IEEE Trans. on Systems, Man, and Cybernetics*, **6**, 769 (1976).
Tribus, M., *Thermostatics and Thermodynamics*, D. Van Nostrand (1961).
Utgoff, P., "Perceptron Trees: a Case Study in Hybrid Concept Representations," *Proc. of AAAI88*, Vol. 2, 601, Morgan Kaufmann (1988).
Weiss, S., and C. Kulikowski, *Computer Systems that Learn*, Morgan Kaufmann (1991).
Winston, P., *Artificial Intelligence*, 2nd ed., Addison-Wesley (1984).

Manuscript received July 26, 1991, and revision received Dec. 2, 1991.