1995

Application of decision trees and multivariate regression trees in design and optimization

Babak Forouraghi
Iowa State University

Follow this and additional works at: https://lib.dr.iastate.edu/rtd
Part of the Computer Sciences Commons

Recommended Citation
https://lib.dr.iastate.edu/rtd/10902

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313-761-4700 800/521-0600
Application of decision trees and multivariate regression trees in design and optimization

by

Babak Forouraghi

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Department: Computer Science
Major: Computer Science

Approved:
Signature was redacted for privacy.

Signature was redacted for privacy.

In Charge of Major Work
Signature was redacted for privacy.

For the Major Department
Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa

1995
# TABLE OF CONTENTS

## GENERAL INTRODUCTION
- Incremental Learning with Decision Trees ........................................ 1
- Decision Trees as Information Filters ................................................ 3
- Regression Trees for Multiobjective Optimization .................................. 5
- Dissertation Organization ..................................................................... 11
- References ......................................................................................... 12

## A DECISION TREE ALGORITHM FOR INCREMENTAL BATCH LEARNING
- Abstract ............................................................................................... 20
- Introduction ......................................................................................... 21
- ID3-Type Algorithms ............................................................................ 23
- Incremental ID3-Type Algorithms ......................................................... 24
- IDea ..................................................................................................... 26
- Empirical Results ................................................................................ 29
- Conclusions ......................................................................................... 34
- References ........................................................................................... 34

## IMPROVING NEURAL LEARNING THROUGH ELIMINATION OF REDUNDANCIES IN TRAINING EXAMPLES
- Abstract ............................................................................................... 37
- Introduction ......................................................................................... 38
- IDea ..................................................................................................... 40
- Improving Neural Learning with IDea .................................................... 42
- Conclusions ......................................................................................... 48
- References ........................................................................................... 49

## LEARNING MATERIALS PERFORMANCE KNOWLEDGE FROM MATERIALS PROPERTY DATABASES
- Abstract ............................................................................................... 52
- Introduction ......................................................................................... 52
- Knowledge Management and Expert Systems ....................................... 53
- Decision Tree Classifiers ..................................................................... 55
- Materials Property Databases ............................................................... 57
- Materials Property Databases ............................................................... 61
GENERAL INTRODUCTION

Inductive learning is a powerful methodology which gives learning systems the ability to acquire classification knowledge from a set of past examples [43]. During the past several decades, a number of algorithms have been successfully implemented in order to address this type of learning. For example, learning systems such as AQ15 [44], CN2 [20], ID3 [53] and its descendent C4 family [54], version space algorithms [34,45], and regression tree techniques of CART [11] and IPRT [70] form general descriptions from a given set of training examples in the form of decision trees or decision rules. The key distinguishing feature of all these algorithms is the fact that they perform a heuristic search through the space of symbolic descriptions to generate a set of inference rules [45].

The main concern of this research effort was with investigation of several issues relating to the use of decision trees and regression trees in diverse areas such as discriminant analysis, multivariate regression analysis, and multi-objective optimization in design. In particular, in regards to the task of acquisition of classification knowledge with decision trees, a new ID3-type incremental learning algorithm called IDea was developed. Our empirical investigations demonstrate that IDea can effectively reduce the spatial complexity of a given set of training examples and hence improve the classification accuracy of a decision tree or neural learner. In regards to regression analysis, we have devised a new methodology whereby identification of Pareto-optimal solutions in a multiple-objective optimization task is greatly facilitated by operations of multivariate regression trees. The following passages will further clarify each of these issues.

Incremental Learning with Decision Trees

Inductive learning systems in general can be categorized as non-incremental or incremental. The main difference between these two types of learning is that non-incremental
learning requires that all training instances be present at the time inference rules are generated. This type of learning is desirable in situations where classification knowledge is to be extracted from static databases. On the other hand, incremental methods maintain and update a set of previously learned theories which may be partially incorrect or incomplete [43]. The primary motivation behind using incremental systems is that classification knowledge may be updated with each new observation, thus sustaining a continual basis for reacting to new stimuli [27].

In order to solve the task of incremental concept acquisition, a number of inductive learning systems have been developed. These include AQ15 [44] which incrementally learns disjunctive concepts from a set of examples, version space algorithm [45] which performs a bi-directional search through the space of hypotheses, and finally, ID4 [65] and ID5 [77] family of algorithms which are essentially incremental variants of the ID3 tree induction algorithm. The common feature among all these algorithms is that they update a set of current concept descriptions one instance at a time. One drawback of algorithms such as version spaces and ID5, however, is that they cannot accurately generalize when training data contain inconsistencies. To remedy this situation, a number of extensions have been proposed. For example, Utgoff suggests extending ID5 by incorporating the chi-squared test for stochastic independence [78]. Also, Hirsh has developed a variation on the version space strategy, called incremental version-space merging, which handles corrupted data [34].

It should be pointed out that much of the effort behind the development of incremental learners has been devoted to the instance-by-instance updating of partially formed theories. In contrast, in an alternate approach called incremental batch learning, concept descriptions are updated from multiple batches of training data containing multiple examples [21]. One of the advantages of batch learning, as noted by Clearwater et al., is that updating theories in noisy domains is more accurate if it is tuned for more than one example at a time. Obvi-
ously, the ability to efficiently generalize in presence of noise is imperative in real-world applications where collected data are not always entirely accurate.

In this work, we introduce a new technique which deals with incremental learning of decision trees in noisy domains. Our method, called IDea (Iterative Dichotomizer extended algorithm), is based on the ID3 algorithm. Furthermore, as an addition to ID3's basic capabilities, it addresses two issues which we believe have not been investigated before. First, IDea uses incremental batch learning, which was previously tested on production-rule-based systems to incrementally update a set of current concept definitions in noisy domains. And second, IDea maintains a set of selected training instances which are used for deriving concept descriptions as more training batches become available. This approach to instance selection is similar to instance-based learning algorithms [3] in that training instances are not indiscriminantly accumulated as they become available. Instead, only specific instances which are deemed as most informative are stored for future processing. Our empirical results show how this careful maintenance of training instances can greatly reduce the storage requirements of an incremental learner without sacrificing classification accuracy.

**Decision Trees as Information Filters**

One of the main concerns while utilizing inductive machine learning techniques such as decision tree algorithms [53] or neural networks [61] is with preparation of an appropriate set of training examples. Unfortunately, in many real-world applications such as detection and classification of flaws in material [18], in the absence of a strong a priori domain knowledge regarding the problem at hand, no universal criteria are known which can help in determination of an appropriate set of training examples. Consequently, learning systems are often inundated with multitudes of examples which often contain redundancies both in the number of training examples presented to the learner and also in terms of the features or at-
tributes over which the classification problem at hand is defined. Elimination of these redundancies, therefore, is the main theme of many theoretical and empirical investigations which attempt to improve efficiency of a machine learning system, especially that of most connectionist networks which are susceptible to slow learning rates.

In terms of relevant theoretical investigations for drawing insights into the phenomenon of learning, in general, and for determining the fitness of training information, in particular, the Valiant Framework [79] has received a great deal of enthusiasm from the artificial intelligence community. In his framework, Valiant essentially provides bounds on the number of training examples required in order to give high confidence that the result obtained by the learning algorithm is probably approximately correct (PAC learning). However, direct application of Valiant's methodology to real-world applications is in its infancy stage and its proposed worst-case lower bounds tend not to be useful in practice [68].

Empirical investigations for reducing computational complexity of neural learning, on the other hand, have examined practical issues which not only relate to the bounds on the number of samples needed for effective learning [42], but also to the underlying architecture of a given network such as the number of hidden layers and/or the number of nodes per hidden layer [64]. From a somewhat different perspective, Sethi's hybrid approach of entropy nets [66,67] primarily uses a training population to induce an ID3-type decision tree. The structure of the induced decision tree is then directly utilized for determining the architecture of a connectionist network which is trained on the original set of training data. Present progress in entropy nets has illustrated the fact that the self-configuration capability of a network achieved through the process of tree-to-network mapping can potentially yield improvements in the classification performance of an inductive learning system.

Mention must be made that most empirical studies to date have emphasized the issue of optimization of connectionist learning in relation to the architecture of neural networks.
While architectural aspects of network design are a crucial precursor to neural learning, the important fact remains that elimination of data redundancies in training samples needs to be addressed in isolation from particular network architectural implementations. The importance of this need is the main motivation behind Wan and Wong's approach [80] where linear classifiers are learned from a subset $S$ of a finite set $X$ of linearly separable vectors. As stated by Wan and Wong, however, the worst-case cost of using their approach is the high computational complexity of solving $O(|X|^3)$ linear programming problems.

In this paper, in an attempt to address the issue of elimination of data redundancies for optimization of inductive neural learning, we introduce a new approach which creates a synergistic relationship between symbolic learning of decision tree algorithms and subsymbolic learning of neural networks. Specifically, we utilize an incremental batch learning decision tree algorithm called IDea [28] in order to reduce the dimensionality of an original training sample space in two ways. First, the number of features or attributes over which the problem is defined is reduced by using the concept of information gain derived from the field of statistical information theory. And second, the incremental batch learning method of IDea continuously discards training information which are deemed as superfluous or unnecessary. After completion of batch learning, the reduced training sample set is fed into a standard, back-propagation network which undergoes a neural learning phase. Our empirical evidence demonstrates that the synergistic relationship created in the aforementioned manner between the two paradigms of symbolic and subsymbolic learning can greatly improve the effectiveness of an inductive neural learning system.

Regression Trees for Multiobjective Optimization

Most engineering design problems involve optimization of several often conflicting objectives in presence of multiple constraints. The literature of multiobjective optimization has
therefore been enriched since its inception in the 1960s by myriads of approaches representing diverse viewpoints and emphases from various disciplines [16]. Generally, the main goal of all these approaches is to optimize vector-valued objective functions where some or all of the objectives are most often noncommensurable. The problem of finding an optimum vector-valued objective is commonly referred to as *vector optimization* in the literature, and solutions of such problem are hereinafter called *noninferior* or *Pareto-optimal* as introduced by Pareto, a prominent economist at the turn of the century [17]. Determination of Pareto-optimal solutions in a given optimization task is essentially tantamount to finding a vector of optimal objectives where an individual objective can be further improved only at the cost of degrading at least one other objective [16].

Traditionally, multiple objective optimization procedures can be grouped into one of two categories: First, vector optimization techniques which rely on conventional mathematical programming (linear or nonlinear), goal programming, utility theory, etc. [35]; and second, statistical approaches such as multivariate analysis of variance (MANOVA) [32] and ellipsoidal design centering techniques [2] which are used extensively in the area of quality control. Regardless of their underlying methodological differences, however, these techniques generate Pareto-optimal solutions which generally lack two crucial characteristics. First, these solutions are represented as points in both the space of design variables (independent parameters) and objective space (dependent parameters). And second, the obtained solutions are rigid in that they do not provide any understanding of the complex nature of the underlying problem which is to be solved. The following passage will clarify these points.

Relating to the first shortcoming of already-available techniques, in many real-world situations, either due to processing limitations or economic factors, it is nearly impossible to pinpoint a singular point as an optimum design vector. For example, setting a beam's scalar design variable *diameter* at 3.3 mm may not be feasible either due to machine processing limitations or high degree of variability which manifests itself during the manufacturing
process (e.g. 3.3 ± 0.9 mm). In fact, the main thrust behind development of the field of robust design is to determine how deviations from an optimum setting of a design vector degrade the overall performance of a system, and consequently, how to make the performance of a system least sensitive to unforeseeable deviations from recommended optimal design settings achieved through mathematical and/or statistical procedures [51].

Secondly, in terms of acquiring an understanding of the nature of the problem at hand, conventional mathematical and statistical techniques are undoubtedly capable of extracting the quantitative knowledge which specifies the static input-output behavior of a system that is to be optimized. In other words, an optimizer provides designers with a recommended set of values for input parameters of interest for which an optimal trade-off situation for objective functions is achieved. These techniques, however, do not and can not extract the knowledge which actually governs the input-output behavior of a system under examination, i.e., these techniques merely indicate what an optimum solution is but can not convey what actually constitutes the optimality of the generated solution. Hence, a particular technique has to be iterated several times under direct supervision of the user in order to obtain how deviations from one particular setting of design variables affect the overall system objectives.

Having stated the two major disadvantages of the traditional approaches to multiple objective optimization, we must now explore what other tools can potentially remedy the situation. As we mentioned, learning the optimization knowledge can have direct benefits to designers. To reiterate, the actual learning of the optimization knowledge in a given task not only allows determination of optimal settings of design variables, but it also allows systematic examination of alternative design scenarios. The learning process, which can be defined as acquisition, assimilation and restructuring of knowledge, has received a great deal of attention from researchers in the field of artificial intelligence (AI) in the past three or four decades [22,68]. In fact, symbolic and subsymbolic learning algorithms have successfully been applied to problems ranging from game playing and logistics to mass spectroscopy and
design of VLSI circuitry [82]. One of the tangible benefits of applying symbolic learning algorithms, in particular, to optimization tasks is the direct acquisition of both the quantitative and qualitative knowledge which governs the behavior of a system that is to be optimized. In response to this observation, the artificial intelligence community has approached this problem by placing special emphasis on development of symbolic search techniques. For example, the MOA* algorithm, although limited in its applicability to real-world problems, was developed as a multiobjective generalization of the heuristic search algorithm A* [10].

Regardless of the range of applicability of MOA*, another artificial intelligence approach which offers a great deal of potential is the paradigm of inductive learning [15]. In an essence, inductive reasoning, first introduced by Bacon in the 1600s [22], is the highly empirical process of drawing conclusions from a given set of observed facts or data obtained through experimentation. Assuming that the input-output components of a system can be appropriately represented in the form of attribute-value design vectors, an inductive learner can discover a highly complex, and perhaps nonlinear, relationship between a set of inputs and output parameters. In fact, the process of symbolic induction (as opposed to numeric induction of subsymbolic, connectionist approaches such as neural networks) itself can be viewed as a form of optimization [60] within which concepts of interest are acquired through numeric/logical processes and are then represented in symbolic forms (e.g., sentences in the first order predicate calculus) which are more in tune with human understanding.

Another paramount issue which needs to be examined in more detail is the methods of representation of optimization knowledge extracted through inductive reasoning. Learning, as we stated before, involves not only acquisition of domain-specific knowledge, but it also requires suitable methods of representing and manipulating the explicated knowledge. Some of most widely-used tools for knowledge representation are propositional calculus, first order logic and semantic networks which include decision trees, frames, and scripts [82].
Among various types of semantic networks, however, decision trees [15] and regression trees [11] have been by far the most extensively-used tools in conjunction with inductive learning. Commonly, decision trees are used in classification tasks where the correct class of an object is predicted and represented in the form of logical expressions involving a vector of inputs which describes that object in terms of its primary attributes. Regression trees, on the other hand, perform piecewise regression of continuous, complex surfaces where each leaf of an induced tree potentially identifies a simple regression subsurface. Clearly, in terms of their applicability to optimization, regression trees, in our opinion, are the most viable method of knowledge representation.

We must emphasize at this point that possibility of employing inductive learning in the area of multiobjective optimization, in general, and utilization of regression trees, in particular, have not been investigated before. The most related development to date is concerned with application of inductive, tree-structured approaches to univariate regression such as classification and regression trees (CART) [11] and inductive partitioning with regression trees (IPRT) [70]. These algorithms are powerful in that not only do they perform ordinary regression, but they also learn regression surfaces by extracting the knowledge that governs the input-output behavior of the model under consideration in the form of regression trees. This form of knowledge acquisition and representation is of utmost importance if regression trees are used for performing optimization tasks. The case in point is that an induced tree essentially represents complex regression surfaces in terms of a number of simpler regression subsurfaces. Detailed examination of these subsurfaces therefore can potentially identify design regions where a product or process response is optimized. Consequently, in addition to pinpointing optimal response regions, tree-structured approaches to optimization offer the advantage of explicating the knowledge that actually constitutes the optimality of the generated solutions. This mode of behavior of regression trees is clearly ad-
vantageous over the traditional, numeric response surface methodology [72] in that it provides a direct means for capturing both the quantitative and qualitative optimization knowledge.

The main disadvantage of techniques such as CART and IPRT lies in the fact that they only address univariate regression analysis. In a typical multiobjective decision-making task, however, the optimization problem at hand consists of several design variables which in turn specify the behavior of a number of responses. For example, in a quality control application, the main objective may be to discover the underlying knowledge which controls the performance of an electric discharge machining (EDM) process in which process variables such as pulse duration and discharge current directly determine several process responses such as electrode wear, surface roughness and metal removal rate [48]. Evidently, in these situations approaches such as CART and IPRT are insufficient due to their single-response limitations.

To summarize, in this work we present a new framework within which multiobjective optimization is accomplished through induction of multivariate regression trees. Furthermore, we present a tree partitioning algorithm which utilizes a number of inductive partitioning criteria based on concepts from statistics and fuzzy logic. Obviously, the choice of using the traditional statistical formulations in this work was instigated by the historic fact that statistics is a firmly established science with many facets which render it a particularly viable tool in many scientific applications. The theory of fuzzy sets [30,36,38], on the other hand, is a more recently developed concept, and it too has proven to be an invaluable tool in a wide array of applications ranging from pattern recognition and clustering to design of digital circuits and relational data bases [49]. In fact, within the context of multiobjective optimization, Bellman and Zadeh's fuzzy approach to optimization [8] has been widely implemented in many engineering structural optimization applications [59]. Therefore, in an attempt to examine the effects of various types of regression-tree partitioning criteria on the
overall learning process which was previously explored only for decision tree algorithms [13], and also, to assess the feasibility of techniques based on fuzzy logic we describe seven splitting rules. Specifically, these include: two statistical decision rules based on dispersion matrices [75], a statistical measure of covariance complexity which is typically used for obtaining multivariate linear models [9], two newly-formulated fuzzy partitioning methods based on Pearson's parametric [32] and Kendall's nonparametric [71] measures of association, Bellman-Zadeh's decision-maximizing fuzzy approach [8] to optimization in an inductive framework, and finally, the multidimensional extension of a measure of fuzzy entropy [38].

Dissertation Organization

This dissertation is organized to include papers that either have been presented and published in refereed conferences or will be submitted to scholarly journals. Specifically, this dissertation includes a general introduction which is followed by six complete papers and a general summary. Each of the six papers is organized as a single, complete document and includes abstract, introduction, literature review, methodology, examples, and conclusion sections.

The first paper included presents a new technique for performing incremental learning with decision tree classifiers and was published in the Proceedings of the Second World Congress on Expert Systems, pp. 830-842. The concern of the second paper is with the improvement of the neural learning of a standard backpropagation neural network by using a decision tree classifier as an intelligent information filter. This paper is submitted to the Third World Congress on Expert Systems. The third paper discusses a new technique for acquiring the relationship between physical properties of materials and their various performance measures and was accepted by the ISMM International Conference on Intelligent Man-
agement Systems. The fourth and fifth papers present a novel methodology for using multivariate regression trees in design and optimization and were published in the Proceedings of the Twelfth AAAI National Conference on Artificial Intelligence, pp. 607-612, and the Proceedings of the Third IEEE International Conference on Fuzzy Systems, pp. 1440-1405, respectively. And finally, the sixth paper included in this work is a complete extension of the last two papers and is submitted to the Journal of Artificial Intelligence Research.

References


A DECISION TREE ALGORITHM FOR INCREMENTAL BATCH LEARNING

A paper published in the Proceedings of the Second World Congress on Expert Systems

Babak Forouraghi, Lester W. Schmerr, Gurpur M. Prabhu

Abstract

To date, Quinlan's ID3 algorithm is the most widely-used decision tree method which has successfully been applied to problems across a variety of domains. One of the drawbacks of ID3, however, is that it cannot directly acquire concepts incrementally.

In this paper, we introduce a new technique which deals with incremental learning of decision trees. Our proposed method, called IDea (Iterative Dichotomizer extended algorithm), is based on ID3. However, as an addition to ID3's basic capabilities, it addresses two issues which have not been investigated before. First, IDea uses incremental batch learning, which was previously tested on production-rule-based systems to update partially formed concept definitions. And second, similar to instance-based learning algorithms, IDea maintains a set of selected training instances which are used for deriving concept descriptions as more training batches become available. Our empirical results show how this careful maintenance of training instances can greatly reduce the storage complexity of an incremental learner without sacrificing classification accuracy. We demonstrate the benefits of our technique with two flaw characterization problems in the area of ultrasonic nondestructive evaluation.
Introduction

Inductive learning is a powerful methodology which gives learning systems the ability to acquire classification knowledge from a set of past examples. During the past several decades, a number of algorithms have been successfully implemented in order to address this type of learning. For example, learning systems such as AQ15 [8], CN2 [3], ID3 [10] and its descendent C4 [11], and version space algorithm [9] form general descriptions from a given set of examples in the form of decision trees or decision rules. The key distinguishing feature of all these algorithms is the fact that they perform a heuristic search through the space of symbolic descriptions to generate a set of inference rules [7].

Inductive learning systems in general can be categorized as non-incremental or incremental. The main difference between these two types of learning is that non-incremental learning requires that all training instances be present at the time inference rules are generated. This type of learning is desirable in situations where classification knowledge is to be extracted from static databases. On the other hand, incremental methods maintain and update a set of previously learned theories which may be partially incorrect or incomplete [7]. The primary motivation behind using incremental systems is that classification knowledge may be updated with each new observation, thus sustaining a continual basis for reacting to new stimuli [5].

In order to solve the task of incremental concept acquisition, a number of inductive learning systems have been developed. These include AQ15 [8] which incrementally learns disjunctive concepts from a set of examples, version space algorithm [9] which performs a bi-directional search through the space of hypotheses, and finally, ID4 [15] and ID5 [17] family of algorithms which are essentially incremental variants of the ID3 tree induction algorithm. The common feature among all these algorithms is that they update a set of current concept descriptions one instance at a time. One drawback of algorithms such as version
spaces and ID5, however, is that they cannot accurately generalize when training data contain inconsistencies. To remedy this situation, a number of extensions have been proposed. For example, Utgoff suggests extending ID5 by incorporating the chi-squared test for stochastic independence [18]. Also, Hirsh has developed a variation on the version space strategy, called incremental version-space merging, which handles corrupted data [6].

It should be pointed out that much of the effort behind the development of incremental learners has been devoted to the instance-by-instance updating of partially formed theories. In contrast, in an alternate approach called incremental batch learning, concept descriptions are updated from multiple batches of training data containing multiple examples [4]. One of the advantages of batch learning, as noted by Clearwater et al., is that updating theories in noisy domains is more accurate if it is tuned for more than one example at a time. Obviously, the ability to efficiently generalize in presence of noise is imperative in real-world applications where collected data are not always entirely accurate.

In this paper, we introduce a new technique which deals with incremental learning of decision trees in noisy domains. Our method, called IDea (Iterative Dichotomizer extended algorithm), is based on the ID3 algorithm. Furthermore, as an addition to ID3's basic capabilities, it addresses two issues which we believe have not been investigated before. First, IDea uses incremental batch learning, which was previously tested on production-rule-based systems to incrementally update a set of current concept definitions in noisy domains. And second, IDea maintains a set of selected training instances which are used for deriving concept descriptions as more training batches become available. This approach to instance selection is similar to instance-based learning algorithms [1] in that training instances are not indiscriminantly accumulated as they become available. Instead, only specific instances which are deemed as most informative are stored for future processing. Our empirical results
show how this careful maintenance of training instances can greatly reduce the storage requirements of an incremental learner without sacrificing classification accuracy.

The remainder of this paper is organized as follows. The next section provides a brief review of the ID3 algorithm and its capabilities. Section 3 discusses previous attempts at extending ID3 for solving the incremental learning task. Section 4 presents IDea's approach to incremental learning. Section 5 introduces two problems in the area of ultrasonic flaw detection and classification where IDea was successfully tested. And finally, the last section summarize the paper.

ID3-Type Algorithms

The ID3 algorithm is an established classification technique today [10]. The basic idea behind ID3's learning algorithm is as follows. During the training phase, the learner is presented with a training set containing the correct classification of an object along with the values of attributes which describe that object. The induction task is to develop classification rules which most accurately determine the class of the object from the values of its attributes. The induced classification rules by the algorithm take the form of a decision tree which essentially contains two types of nodes: nonterminal nodes and terminal nodes. Nonterminal nodes contain decisions which simply test values of the attributes over which the object is defined. Terminal or leaf nodes correspond to decision outcomes or classification labels. A branch of the tree then, starting from the root and ending in a leaf node, is a compound decision made of the conjunction of several simpler decisions.

ID3's tree induction mechanism is a top-down recursive algorithm. Initially, all the training examples are placed at the root of a tree. Subsequently, an attribute and attribute value are selected which best partition the training set into a number of subsets. The best at-
tribute is selected according to the information entropy criterion which measures the randomness or impurity in a given set of examples. Hence, starting with maximum uncertainty at the root of the tree, the induction task recursively expands the tree from the top down as to minimize the randomness in the training subsets created at each node of the tree. The tree expansion halts when a leaf node contains training examples which belong to the same class or when some other stopping criterion is met.

The practical importance of IDS lies in the fact that it can handle noisy input data and unknown attribute values. The noise handling mechanism is called pessimistic error pruning. Basically, after the tree induction task is complete, the pruning process starts by examining the induced tree from the bottom up. A subtree is pruned back and replaced by one of its descendant nodes if the resulting substitution improves the overall accuracy of the induced classifier based on the information contained in the training set. Eventually, the pruning process stops when no further improvements are possible [11-13]. Furthermore, ID3 employs a probabilistic approach to cope with unknown attribute values during the training and testing phases [14].

To reiterate, ID3's ability to deal with noisy and incomplete input data is quite appealing in the real-world applications where such anomalies are pervasive. A potential limitation of ID3, however, is its lack of ability to directly handle incremental concept acquisition. In the next section, we provide a summary of previous investigations aimed at overcoming this limitation.

**Incremental ID3-Type Algorithms**

The task of incremental learning of decision trees can essentially be solved by: (1) brute-force reconstruction, (2) informed reconstruction, or (3) restructuring or revision.
In the brute-force approach, a decision tree is induced on a global training set which is augmented by new training examples as they become available. This technique, obviously, has the limitation that the size of the training set, and consequently, the computational complexity, can grow rapidly in situations where new observations become available frequently over time.

The informed reconstruction approach implemented in ID4 [15] examines an induced tree after a new training instance has arrived. If an attribute test in the tree no longer provides maximum information gain, that attribute is discarded and a new attribute test is created. This implies that a new subtree for the newly selected attribute must be created each time the maximum information principle is violated. As Utgoff notes, ID4 is inadequate in that a tree may never reach stability during incremental reconstruction. For example, if the attribute at the root of the tree is discarded after examining each new training instance, then the entire tree has to be reconstructed. This type of instability or thrashing of a tree can render certain ID3-learnable concepts unlearnable by ID4 [18].

In an attempt to eliminate ID4's potential drawback, Utgoff introduced ID5 algorithm which uses a "pull-up" procedure [17]. Essentially, ID5 may restructure a decision tree for every newly available training instance. But, instead of discarding the subtree below an invalid decision node and losing previous training efforts, ID5 reshapes the tree by first finding a new best test attribute and then pulling that best attribute up from below. It is shown that ID5's performance is superior to that of ID4, and also that of a brute-force ID3, where a new decision tree is built from the entire set of training instances observed thus far. One drawback of ID5, nevertheless, is that it if the training instances are corrupted with noise or are unknown, as it is the case in many real-world applications, ID5 needs extensions that would let it deal with these limitations [18].
Presently, ID4 and ID5 techniques compromise the bulk of the research efforts that have dealt with the task of incremental construction and/or revision of ID3-type decision trees. In the following section, we present our proposed method, IDea, which approaches incremental learning in a new unified framework.

IDea

The main motivation behind conception of IDea was two-fold. First, the incremental learning technique offered by ID5, in its present form, was not adequate for our application area where noisy data are pervasive. And second, the traditional instance-by-instance learning, used by ID4 and ID5 both, may not always perform satisfactorily when drawing generalizations from noisy instances. This second point will be clarified in the following passage.

Clearwater et al. previously introduced an alternate class of inductive learning called incremental batch learning which was tested on the production-rule-based system Meta-DENDRAL [4]. The major consideration in this work is that new training instances arrive in batches over the lifetime of a learning system. Batch learning, therefore, can be viewed as a midpoint between the extreme approaches of single-batch learning of ID3 and instance-by-instance learning of ID4 and ID5 (batches of size one). The appealing feature of this type of learning, as noted by Clearwater et al., is that updating partially formed hypotheses in noisy domains is more accurate if it is tuned for more than more than one example at a time. An important remark to be made here is that this type of learning is consistent with ID3’s own internal learning mechanism. ID3 induces a tree on a random subset of the original training set called the working set. The resulting tree is then tested on the remainder of the unused cases, and some portion of the misclassified cases is then added to the working set. Batch learning primarily uses the same method with the exception that all of the misclassified cases result in concept modifications.
In Section 3, we stated that incremental learning can be approached in several ways. For example, ID4 and ID5 algorithms focus on partial reconstruction and revision of already induced trees, respectively, in order to save previous training efforts. IDea, on the other hand, uses the reconstruction technique each time a new batch of training instances arrive. However, as opposed to the single-batch ID3 algorithm where all training instances are aggregated in a current training set, we use an informed instance selection technique which can greatly reduce storage requirements while maintaining classification accuracy comparable to, and in some instances better than, that of a brute-force ID3 algorithm.

The instance selection algorithm for incremental batch learning implemented in IDea is based on concepts from instance-based learning (IBL). In this type of learning, which is an extension of the nearest neighbor type of pattern classification, training instances implicitly represent a set of concept descriptions [1]. In other words, classification predictions in this framework are based solely on training instances rather than on a set of abstractions derived from the training instances. In their early stages, IBL algorithms were computationally expensive because they saved all the training instances. Subsequently, in an attempt to reduce storage requirements, Aha et al. proposed saving only informative instances which were identified as those misclassified by the learner.

The key idea, therefore, behind IDea's instance selection is as follows. Assume that a new batch of training instances arrives. IDea tests the already induced tree on the new batch of observations. If all the training instances are correctly classified, then we conclude that concept descriptions are adequate and need no further modifications. If any examples are misclassified, they are added to the current pool of representative exemplars over which a new tree is induced. As the learning process progresses, the current set of examples expands to include only typical cases along with atypical cases. It must be stated that this approach is different than Zhang's hybrid system [19] where typical cases are represented with a set of
rules, and atypical cases, which are viewed as exceptions to the concept, are matched against future events using similarity-based functions used in IBL algorithms. In an essence, Zhang's solution first uses induced rules to capture concept regularities and then forms irregular concept boundaries by using exceptions. IDea, on the other hand, is able to capture regular and irregular concept boundaries in a unified manner through its tree-pruning process.

Having discussed the key issues pertaining to IDea's framework for learning, Fig. 1 formally sketches IDea's algorithm. Here, the set INSTANCES is initially empty. As training examples arrive, INSTANCES grows to include examples which successfully contribute to concept identification. Traditionally, once the generalizations are formed, these informative instances remain buried in a decision tree without any further direct use. IDea, however, externally stores and maintains a global training set which is modified during the learning process.

/* Assume a new training batch has arrived */

Retrieve the current set INSTANCES;
Retrieve tree $T$ which was previously induced on INSTANCES;
Let $TB :=$ the new training batch;
FOR (all examples $E$ in $TB$) {
    IF ($E$ is misclassified by $T$) THEN
        Expand INSTANCES to include $E$;
    }
IF (INSTANCES was modified)
    Induce a new tree $T_{new}$ on INSTANCES and save it;

Fig. 1. IDea's learning algorithm
The main advantage of this scheme is that some of the difficulties associated with capturing concept drifts can be alleviated. In many real-world applications, concept drifts are naturally occurring phenomena which require adjustments in current concept descriptions over time. In absence of any mechanism such as the one used in IDea, a learning system can capture concept drifts in two ways: (1) by the brute-force approach of saving all training instances which is computationally expensive, or (2) by some other mechanism such as ID5's instance-by-instance tree revision technique which needs further modifications to cope with noisy input data. The simple and intuitive solution offered by IDea, however, relies on the fact that some of the difficulties in incremental learning can be circumvented by a careful selection of training instances used for learning. The empirical results provided in the next section will better justify this point.

**Empirical Results**

In this section, we report on the empirical behavior of IDea when applied to two flaw characterization problems in the area of ultrasonic nondestructive evaluation. Essentially, ultrasonic evaluation is a non-intrusive technique whereby ultrasonic pulses are launched into a part for assessing the part's structural integrity. Different types of anomalies present in the returning ultrasonic pulses then indicate existence of different types of flaws such as crack or non-crack.

The first application we present is concerned with the detection of hard-alpha inclusions in titanium alloys which often initiate cracking in aircraft components [2]. The data set used contained 230 examples which was split randomly into a training set and testing set each containing 115 cases. The input data in both sets were characterized by 6 real-valued, noisy attributes and 2 classes of flaw and noise.
The second application described involves the problem of flaw classification. The input data consisted of 239 examples which were originally generated from samples provided by the Westinghouse Corporation [16]. Again, the input data was split randomly into a training set and a testing set containing 120 and 119 examples, respectively. The input data were characterized by 14 real-valued, noisy attributes and 2 classes of crack and non-crack.

Before reporting on the obtained results, two remarks need to be made. First, we use the tuple \((PY, PN, TC)\) to report on the accuracy of an induced tree for a given concept where:

- \(PY\): percent of examples classified as having a performance parameter present given that the performance parameter is present (higher \(PY\) implies higher accuracy)
- \(PN\): percent of examples classified as having a performance parameter present given that the performance parameter is not present (lower \(PN\) implies higher accuracy)
- \(TC\): tree complexity which is measured by a tree’s number of terminal leaves

And second, in each experiment the training set was split randomly into various number of batches in five separate runs in order to simulate incremental batch learning. Each \((PY, PN, TC)\) point in Figures 2 and 3, hence, is averaged over five statistically independent runs.

**Hard-alpha Detection**

Fig. 2 illustrates the behavior of IDea when tested on the hard-alpha inclusion detection problem where the original training data were split randomly into 2, 5, 10, and 20 batches. Note that when operating on one batch, IDea’s performance subsumes that of a brute-force
ID3 using the entire training set. As it is evident, IDea was able to reduce the training set size from 115 examples (one batch) to 63 (2 batches), 35 (5 batches), 24 (10 batches), and finally, 22 examples (20 batches) while maintaining an identical PY measure of 73% and comparable (PN, TC) measures. This implies that IDea can be trained faster than a single-batch ID3 and still produce comparable, and in few instances, superior results to those produced by the ID3 algorithm.

Fig. 2. Hard-alpha detection (concept flaw)
Another important factor which was found to greatly affect IDea’s batch learning is the batch size. Essentially, as the number of batches increases, the number of examples contained in each batch decreases. Continuing in this manner, batch learning eventually subsumes instance-by-instance learning when the batch size becomes one. In this experiment, the largest acceptable number of batches was empirically determined to be 20 since the performance degradation became notable for number of batches greater than 20. This result is in complete agreement with Clearwater’s statement that incremental learning in noisy domains is more accurate when it is tuned for more than one example at a time.

**Flaw Classification**

The obtained results for the second example, the flaw classification problem, follow the basic trend exhibited in the first example. As depicted in Fig. 3, IDea was able to reduce the training set size from 120 examples (one batch) to 75 (2 batches), 57 (5 batches), and 53 examples (10 batches). The \( PY \) measures in all cases remained at 79% while \( (PN, TC) \) measures fluctuated at comparable levels. Here, the largest number of batches beyond which IDea’s performance began to deteriorate was determined to be 10. Also, note that the tree complexity measure, \( TC \), was comparable for the brute-force ID3 and IDea which proves that reduction of the training set maintains classification accuracy without necessarily producing more complicated and larger decision trees.

In closing, it must be underlined that our experimental results show that IDea’s mechanism for selection of informative training instances facilitates efficient incremental learning. This was illustrated by the fact that, when compared to a single-batch ID3, IDea attained comparable to slightly superior results from smaller number of training instances. Also, the reported performance measures were quite favorable to those reported on other established classification techniques tested on the same data sets. In the case of the hard-alpha inclusion
Fig. 3. Flaw classification (concept crack)

detection problem, for example, a standard backpropagation neural network achieved \((PY, PN)\) measure of (73\%, 6\%) [2] which is comparable to IDea's (73\%, 9\%) accuracy on 5 training batches. For the second example, IDea's (79\%, 21\% to 24\%) performance in all cases was slightly superior to that of a probabilistic neural net which achieved (77\%, 22\%) accuracy [16].
Conclusions

In this paper, a new approach to incremental construction of ID3-type decision trees was described. The main motivation behind conception of our proposed solution, called IDea, is that previous solutions to incremental learning such as ID4 and ID5 algorithms were seen inadequate due to their instance-by-instance type of approach to incremental learning. In contrast, IDea uses incremental batch learning which can essentially be viewed as learning from multiple batches containing multiple examples. One advantage of batch learning is that learning in noisy domains is shown to be more accurate if it is tuned for more than one example at a time.

Furthermore, we demonstrated how IDea’s storage requirements, and hence, its computational complexity can greatly be reduced by adopting instance selection techniques commonly used in instance-based learning (IBL) algorithms. The experiments we conducted show that IDea’s performance is comparable to, and in some instances better than, that of single-batch ID3 algorithm while using notably fewer training instances.

References


IMPROVING NEURAL LEARNING THROUGH ELIMINATION OF REDUNDANCIES IN TRAINING EXAMPLES

A paper submitted to the Third World Congress on Expert Systems

B. Forouraghi, Gurpur M. Prabhu, Lester W. Schmerr

Abstract

In this paper we introduce a novel technique for improving the inductive neural learning of a standard, back-propagation connectionist network. The improvement in learning is essentially accomplished by using an incremental ID3-type decision tree algorithm called IDea as an intelligent information filter for reducing the dimensionality of a given set of training examples. In particular, given an original population of training data, IDea effectively filters out the redundancies that may occur in the given population along the two dimensions of instance space and feature space. The reduced training information, which is subsequently utilized to train a neural network, is shown to considerably reduce the overall training time while maintaining or improving the classification accuracy of the connectionist inductive learner. The importance of our technique lies in the fact that in most real-world applications, in the absence of a strong a priori domain knowledge regarding exact characterizations of a problem of interest, a machine learning system is often inundated with a myriad of training examples with inherent redundancies which in turn tend to degrade the learning efficiency of the learner.

We highlight the advantages of our proposed technique with an example regarding the detection and classification of hard-alpha inclusions in titanium alloys which has proven to be a difficult problem in the field of nondestructive evaluation of materials.
Introduction

One of the main concerns while utilizing inductive machine learning techniques such as decision tree algorithms [6] or neural networks [8] is with preparation of an appropriate set of training examples. Unfortunately, in many real-world applications such as detection and classification of flaws in material [2], in the absence of a strong a priori domain knowledge regarding the problem at hand, no universal criteria are known which can help in determination of an appropriate set of training examples. Consequently, learning systems are often inundated with multitudes of examples which often contain redundancies both in the number of training examples presented to the learner and also in terms of the features or attributes over which the classification problem at hand is defined. Elimination of these redundancies, therefore, is the main theme of many theoretical and empirical investigations which attempt to improve efficiency of a machine learning system, especially that of most connectionist networks which are susceptible to slow learning rates.

In terms of relevant theoretical investigations for drawing insights into the phenomenon of learning, in general, and for determining the fitness of training information, in particular, the Valiant Framework [12] has received a great deal of enthusiasm from the artificial intelligence community. In his framework, Valiant essentially provides bounds on the number of training examples required in order to give high confidence that the result obtained by the learning algorithm is probably approximately correct (PAC learning). However, direct application of Valiant's methodology to real-world applications is in its infancy stage and its proposed worst-case lower bounds tend not to be useful in practice [11].

Empirical investigations for reducing computational complexity of neural learning, on the other hand, have examined practical issues which not only relate to the bounds on the number of samples needed for effective learning [4], but also to the underlying architecture of a given network such as the number of hidden layers and/or the number of nodes per hid-
den layer [9]. From a somewhat different perspective, Sethi’s hybrid approach of entropy nets [10] primarily uses a training population to induce an ID3-type decision tree. The structure of the induced decision tree is then directly utilized for determining the architecture of a connectionist network which is trained on the original set of training data. Present progress in entropy nets has illustrated the fact that the self-configuration capability of a network achieved through the process of tree-to-network mapping can potentially yield improvements in the classification performance of an inductive learning system.

Mention must be made that most empirical studies to date have emphasized the issue of optimization of connectionist learning in relation to the architecture of neural networks. While architectural aspects of network design are a crucial precursor to neural learning, the important fact remains that elimination of data redundancies in training samples needs to be addressed in isolation from particular network architectural implementations. The importance of this need is the main motivation behind Wan and Wong’s approach [13] where linear classifiers are learned from a subset $S$ of a finite set $X$ of linearly separable vectors. As stated by Wan and Wong, however, the worst-case cost of using their approach is the high computational complexity of solving $O(|X|^3)$ linear programming problems.

In this paper, in an attempt to address the issue of elimination of data redundancies for optimization of inductive neural learning, we introduce a new approach which creates a synergistic relationship between symbolic learning of decision tree algorithms and subsymbolic learning of neural networks. Specifically, we utilize an incremental batch learning decision tree algorithm called IDea [3] in order to reduce the dimensionality of an original training sample space in two ways. First, the number of features or attributes over which the problem is defined is reduced by using the concept of information gain derived from the field of statistical information theory. And second, the incremental batch learning method of IDea continuously discards training information which are deemed as superfluous or unnecessary. Af-
ter completion of batch learning, the reduced training sample set is fed into a standard, back-propagation network which undergoes a neural learning phase. Our empirical evidence demonstrates that the synergistic relationship created in the aforementioned manner between the two paradigms of symbolic and subsymbolic learning can greatly improve the effectiveness of an inductive neural learning system.

The remainder of this paper is organized as follows. Section 2 presents an overview of IDea's incremental batch learning mechanism for reducing the dimensions of a set of training data. Section 3 discusses the results of applying IDea to the problem of classification of hard-alpha inclusions in titanium alloys and reports on improvements achieved both in symbolic learning of IDea itself and also that of a neural network. And finally, Section 4 is the summary and conclusions.

IDea

Symbolic inductive learning is a widely-used technique for explicating classification knowledge from a given set of training samples[5]. As an example, ID3-type decision tree algorithms [6] are used extensively by the virtue of the fact that in contrast to the black box approach of neural networks, decision tree algorithms allow direct examination of the classification knowledge extracted during the induction process in the form of sentences in symbolic logic. One of the shortcomings of ID3-type algorithms, however, is their lack of ability to deal with learning in dynamic environments where new training information arrives in multiple batches. An extension of ID3-type algorithms called IDea [3] was specifically designed to handle the problem of incremental batch learning in an intuitive manner. Figure 1 depicts IDea's incremental learning algorithm.
/* Assume a new training batch has arrived */

Retrieve the current set INSTANCES (Initially Empty);
Retrieve tree T which was previously induced on INSTANCES;
Let TB := the new training batch;
FOR ( all examples E in TB ) {
    IF ( E is misclassified by T ) THEN
        Expand INSTANCES to include E;
    }
IF ( INSTANCES was modified )
    Induce a new tree $T_{\text{new}}$ on INSTANCES and save it;

Figure 1. IDea’s main algorithm

As shown above, incremental batch learning is quite intuitive in nature. Essentially, IDea always maintains a global set of training examples $S$ with its corresponding induced decision tree $T$ which is subsequently tested on new batches of training examples as they arrive over time. If $T$ correctly classifies all the new examples, then $T$ is considered consistent with $S$. Otherwise, $S$ is augmented with all the misclassified cases from the new batch and a new tree is induced over $S$.

The main motivation behind the development of IDea, aside from incremental learning issues, was the fact that inductive learning traditionally proceeds in one of two possible spaces: (1) space of instances, and (2) the space of hypotheses [7]. For instance, an ID3 algorithm attempts to form generalizations in the space of hypotheses using sentences of symbolic logic in order to capture the underlying classification knowledge. Instance-based learning algorithms [1] or nearest-neighbor type statistical techniques [14], on the other hand, do
not form any abstractions over the given data but rather rely on the training instances themselves to perform the necessary classification process. IDea’s approach, consequently, can be considered as a scheme which operates both in the space of hypotheses and instances. One of the direct implications of such an approach is that the space of instances can potentially be modified and updated in order to eliminate redundancies.

**Improving Neural Learning with IDea**

A potential application of IDea was previously demonstrated in the area of detection and classification of hard-alpha inclusions in titanium alloys [2]. Briefly, this problem involves classification of hard-alpha anomalies (flaws vs. non-flaws) over a training/testing set of 115 examples. Each training/testing vector, correctly classified as flaw (15 cases) or non-flaw (100 cases), essentially describes quantitative measurements of ultrasonic signals over 6 continuous features such as number of zero crossings, mean, absolute mean, variance, skewness, and kurtosis of digitized ultrasonic signals relating to various titanium blocks which either contained or did not contain flaws.

Before proceeding with exact details for filtering out the inherent redundancies in a given training data set, we must briefly mention our method of analysis for determining the goodness of an induced tree. Basically, once a tree is induced over a set of training examples, it is then tested on a testing set. Once the testing process is complete, two error measures which are analogous to Type I and Type II error measures were computed for each classification label (flaw and non-flaw) as follows:

1. **PY**: refers to percent of examples classified as having a performance parameter (such as flaw or non-flaw) present given that the parameter is present
2. \( PN \): is percent of examples classified as having a performance parameter present given that the parameter is not present

3. \( TS \): the size of an induced true in terms of its number of leaves (terminal nodes)

Note that quantities \( PY \) and \( PN \) are drawn from different populations so their sum for a particular performance criterion is not necessarily 100% but rather:

\[
(1) \quad PY(\text{flaw}/\text{non-flaw}) + PN(\text{non-flaw}/\text{flaw}) = 100\%
\]

Furthermore, the main rationale behind choosing \( TS \) as a measure of tree performance was to ascertain if reducing the size of training sample space would in any way alter or make more complicated the nature of extracted hypotheses in the hypothesis space. Therefore, measure \( TS \) can effectively report on the size of an induced tree, and consequently, indicate how complex explicited symbolic classification rules actually are.

Our empirical procedure for determining and eliminating redundancies in the training examples was performed in two stages. First, in terms of deciding which of the original six features offered maximal discrimination power between the two classes of flaw and non-flaw, a decision tree was induced over various random subsets of the 115 training examples. In all cases, the produced decision trees with maximum \( PY \) (73%) and minimum \( PN \) (17%) measures consistently used the two features of kurtosis and variance as depicted in Figure 2.

The second stage of optimization, viz. minimization of the training sample complexity while maintaining classification accuracy of at least [73%, 17%], was then simulated via incremental batch leaning of IDea. The original training set was randomly divided into 2, 5, 10
Figure 2. An example of an induced tree (TS = 3)
and 20 batches where each batch respectively contained roughly 57, 23, 11, and 5 examples. The IDea algorithm for incremental batch learning was subsequently applied to each of these divisions of the original population of training instances, the results of which are illustrated in Figure 3.

![Figure 3. Results of incremental batch learning with IDea](image-url)
As shown in Figure 3, the original data set with 115 examples produces a tree with classification accuracy of [73%, 17%]. Applying IDEa, however, to 2, 5, 10, and 20 batches enables the classification accuracy measure \( PY \) to remain at the 73% level while \( PN \) measure takes the values 12%, 9%, 20%, and 15%, respectively. The interesting point is that IDEa effectively reduces the size of training set in each of these cases from the original 115 examples to 65 examples (2 batches), 35 examples (5 batches), 24 examples (10 batches), and finally 22 examples (20 batches). It must be mentioned that the tree generated on 5 batches of data with only 35 examples produced the best classification accuracy of [73%, 9%] while relying only on the two features of kurtosis and variance. Consequently, we can empirically determine not only what subset of the original six features contains the most discrimination power, but we can also pinpoint 35 examples out of the original 115 examples which offer a comparable \( PY \) measure of 73% along with a slightly more accurate \( PN \) measure of 9%.

In terms of improving the neural learning of a standard back-propagation network, the reduced training examples along the two dimensions of the kurtosis and variance can be used for the training process. Furthermore, in order to ascertain effects of reduced dimensionality on the behavior of a neural learner, a network must also be trained on the original set of 115 examples along the dimension of six features which were described in the previous section. Table 1 summarizes these results.

As results in Table 1 demonstrate, a connectionist network without using IDEa as an intelligent information filter produces the classification accuracy of [73%, 6%] which is slightly superior to that of an ID3-type decision tree operating on the same data. However, as shown in Figure 2, reducing the dimensionality in the space of training instances (case of 5 batches) not only improves the effectiveness of neural learning to [80%, 0%] but also benefits the decision tree algorithm itself by decreasing its \( PN \) metric from 17% to 9%. Obvi-
ously, the symbiotic relation created in this manner has potential advantages for both the symbolic and subsymbolic learners.

Another point that needs to be emphasized at this point is that the nature of decision tree algorithm is quite deterministic in that, given a set of $X$ examples, a tree is induced in a finite number of steps which is $O(A \cdot X^2)$ where $A$ is the size of attribute or feature space.

Table 1. Classification accuracy of a neural network with and without IDea

<table>
<thead>
<tr>
<th>Classification Technique</th>
<th>Classification Accuracy [PY, PN]</th>
<th>Number of Training Instances Needed</th>
<th>Number of Attributes Needed (out of 6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID3</td>
<td>[73%, 17%]</td>
<td>115</td>
<td>2</td>
</tr>
<tr>
<td>IDea</td>
<td>[73%, 9%]</td>
<td>35</td>
<td>2</td>
</tr>
<tr>
<td>A Standard Backpropagation Neural Network</td>
<td>[73%, 6%]</td>
<td>115</td>
<td>6</td>
</tr>
<tr>
<td>IDea and a Standard Neural Network</td>
<td>[80%, 0%]</td>
<td>35</td>
<td>2</td>
</tr>
</tbody>
</table>

The amount of time required for training a neural network, on the other hand, is nondeterministic and is based on the gradient-descent technique [8]. Therefore, one of the major drawbacks of employing neural networks is their slow rate of convergence, and hence, learning. In our study, in order to address this issue, the absolute training periods (measured in seconds) along with the number of iterations needed to achieve convergence were computed for the overall classification task. These measures appear in Table 2.
### Table 2. Convergence timings

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of Training Examples</th>
<th>Number of Iterations for Convergence</th>
<th>Convergence Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Backpropagation Neural Network (BPNN)</td>
<td>115</td>
<td>10,000</td>
<td>220</td>
</tr>
<tr>
<td>BPNN + IDea (2 Batches)</td>
<td>65</td>
<td>4,000</td>
<td>88</td>
</tr>
<tr>
<td>BPNN + IDea (5 Batches)</td>
<td>35</td>
<td>3,500</td>
<td>77</td>
</tr>
<tr>
<td>BPNN + IDea (10 Batches)</td>
<td>24</td>
<td>900</td>
<td>20</td>
</tr>
<tr>
<td>BPNN + IDea (20 Batches)</td>
<td>22</td>
<td>850</td>
<td>19</td>
</tr>
</tbody>
</table>

In closing this section, mention must be made that in a modest training set of size 115 with only 6 features, the savings in training effort are quite noticeable. For example, average training time of 220 seconds for extracting classification information from the original training set can be effectively reduced by a factor of 2.85 to 77 seconds in the case of running IDea only on 5 batches. It is expected that these savings will be even more considerable while working with problems which are defined in considerably larger instance and feature spaces.

### Conclusions

In this paper we introduced a new technique for reducing the dimensionality of a given set of training examples along the two dimensions of instance space and feature or attribute
space. The rationale behind development of our technique lies in the fact that in most real-world applications, in the absence of any universal criteria for preparation of appropriate training information, inductive machine learning systems are often inundated with multitudes of examples which can contain superfluous information. The presence of these data redundancies in turn can degrade the performance of an inductive learner such as a neural network. Our proposed methodology here effectively eliminates such redundancies by creating a synergy whereby incremental batch learning of an inductive decision tree algorithm is utilized to select a subset of an original set of training data which is deemed as most informative. Our empirical results on the problem of classification/detection of hard-alpha inclusions in titanium alloys indicate that inductive neural learning of a standard connectionist network can generally benefit from such data reductions in two ways. First, in terms of improvements in the classification accuracy of the inductive learner, and second, in terms of savings in the amount of time that is needed to train the network.

In terms of future directions, a noteworthy effort would be to investigate, within a unified framework, the relationship between our empirical results concerning the lower bound on the number of training examples needed to learn classification knowledge with that of theoretical analyses offered by Valiant's PAC learning concept. Work is also under way for preparation of and subsequent experimentation on larger data sets relating to various classification tasks within the field of nondestructive evaluation.

References


Abstract

Assessment of materials performance in engineering design and manufacturing is a complex process which requires significant knowledge of scientific and engineering principles. In a typical design application, for instance, determination of how a material's performance indices such as fatigue resistance, corrosion resistance and formability are affected by its physical properties is a laborious task. Expert system technology has been able to partially address this issue by transferring domain specific expertise and heuristics from human experts into knowledge-based systems which can precisely communicate materials knowledge to their users. Traditional expert systems, however, are limited from the standpoint that acquisition of their needed operational knowledge from human experts, who may not even exist in some particular applications, is a time-consuming process which is at best subjective and prone to error. Furthermore, availability of new materials information requires updating of these systems which is also an extensive undertaking.

In this paper, we present a new approach where the information obtained from a materials database is translated into operational knowledge which is readily made available to designers. In particular, we demonstrate how an ID3-type decision tree learning system can extract such knowledge from a commercially available property database.
Introduction

One of the main concerns in engineering design is with determination of how a material's performance is affected by its physical behavior. For instance, performance indices such as fatigue resistance and formability are directly controlled by properties such as hardness, tensile ultimate strength and tensile yield strength. Unfortunately, understanding the complex relationship that exists between various factors such as the processing characteristics and material properties is a process that demands considerable expertise on the designer's part who may not be an expert process or analysis engineer.

Expert system technology [7] has been able to address this issue by synthesizing the needed materials knowledge and heuristics derived from domain expertise and transferring that knowledge into computer programs that can easily be used as problem solving tools by users. For instance, EXPSSC [26] or Harwell and NPL's corrosion expert systems [4], EXENAC for diagnosing the possibility of stress corrosion cracking in stainless steel materials [14] and an advanced ceramics selector program for assisting in design of gas-fueled heat exchangers [5] are all prime examples of the wide-spread use of expert systems.

The main issue in developing expert systems for selection of materials involves the method of transfer of the appropriate knowledge from human experts into computer programs. Typically, an expert's knowledge is acquired through a series of interviews after which it is codified or hardwired in an expert system's knowledge base [23]. Verbal communication with human experts, nonetheless, is generally a time-consuming task which is at best subjective and prone to error [11]. Furthermore, in many problem domains experts either may not exist or they may be unable to communicate their collective knowledge which is a more complex representation of their past experiences. In these cases, therefore, it is crucial to have the capability to derive the needed materials performance knowledge from
alternate sources of information such as property databases [2]. For instance, commercially available databases such as Mat.DB or M/VISION contain large quantities of data in regards to properties and performance indices of a variety of materials [18]. Translation of these types of data into domain-specific generalizations is the main thrust behind development of the field of machine learning (ML).

ML is an integral component of the science of artificial intelligences which includes other significant developments such as theorem proving, robotics, natural language processing and expert systems technology. Furthermore, ML has grown to the point that it encompasses a number of somewhat overlapping paradigms such as inductive learning, explanation-based learning, genetic algorithms and neural networks [8]. Inductive learning, in particular, is an active research area whose aim is with automatic acquisition of knowledge from rich sources of information. Inductive learning systems, to name a few, include ID3-type decision tree classifiers [22], regression trees [6], instance-based learning algorithms [3] and neural networks [13]. Regardless of their underlying working principles, however, these techniques learn from prototypical training examples which are analogous to an expert’s past problem solving experiences. In the case of machine discovery of the relationships between a material’s performance indices and its properties, henceforth, a materials database is the perfect source of information for an inductive learner.

In this paper therefore we present a new approach where a property database is used as the primary source of information for a machine learning system. In particular, we demonstrate how IDea [12] which is essentially an ID3-type decision tree learning algorithm [22] can extract the implicitly stored materials knowledge from a property database and explicate it in the form of generalizations that can easily be examined by designers. One of the benefits of IDea, aside from its ability to deal with missing property values which occur often in databases, is that it can incrementally learn new concepts with arrival of new materials infor-
mation over time. This is in contrast with traditional expert systems where new information has to be incorporated into the knowledge base through manual intervention.

The remainder of this paper is organized as follows. Section 2 is a brief introduction to expert systems and decision tree classifiers. Section 3 presents some of the commercially available materials databases along with our choice of the database in this work. Section 4 discusses our results relating to explication of knowledge pertaining to relationship between physical properties of materials and their performance. And finally, Section 5 summarizes the paper.

Knowledge Management and Expert Systems

Expert systems are computer programs which represent knowledge and problem-solving capabilities of experts in a variety of decision-support environments. Since their inception in the 1970's, expert systems have been successfully used in a wide array of industrial and business applications ranging from monitoring of spacecraft telemetry to flight scheduling [17]. In fact, one of the early expert systems called MYCIN was the first computer program which could effectively diagnose infectious blood diseases with performance levels comparable to that of human experts [7].

One of the basic components of an expert system which allows representation of external, domain-specific knowledge is the knowledge base [23]. The knowledge base is essentially a collection of IF-THEN rules representing the system's current knowledge regarding a specific task. For example, an ultrasonic flaw detection expert system may contain the following rules:

**Rule 1:** IF (Kurtosis of Detected Signal < 0.19) THEN Signal Source is Noise
Rule 2:  IF [(Kurtosis of Detected Signal >= 0.19) AND (Variance of Detected Signal is >= 0.23)]

THEN  Signal Source is Flaw

Traditionally, acquisition of knowledge in the form of rules is performed through interviews with domain experts. Extracting knowledge from human experts through verbal communication however is a process which is time-consuming, complex and prone to error. Also, in many applications human experts may not be able to communicate their past problem solving experiences in the form of IF-THEN rules [11]. The latter case can best be illustrated by the problem of detection of hard-alpha inclusions in titanium alloys [9]. Hard-alpha flaws are microstructure regions contaminated by inclusion of nitrogen or oxygen and are potentially dangerous in that they can cause cracks in aircraft components made of titanium. Unfortunately, detection of such anomalies is complicated by the lack of any strong a priori domain knowledge about precise characteristics of hard-alpha inclusions. The only remedy in these cases is to predict outcome of an event (presence of hard-alpha flaw) by using a set of past examples which implicitly typify an expert’s knowledge about the problem at hand. This mode of reasoning is called inductive learning and is one of the major thrusts behind development of the field of machine learning [25].

Briefly speaking, learning is the essence of human and artificial intelligence and can be viewed not only as the means for acquiring new knowledge but also as an instrument for refining old knowledge. Some of the widely used inductive, machine learning techniques today are decision tree classifiers such as ID3-type algorithms [22] and CART [6], neural networks [13] and instance-based, nearest-neighbor type of approaches [3]. Regardless of their underlying working mechanisms, these approaches learn to solve various classification tasks by capturing domain-specific knowledge from a set of previous typical examples. In particu-
lar, decision tree algorithms, which will be emphasized hereinafter, have been widely used in diverse areas such as radar signal classification, remote sensing and speech recognition [24] mainly because of their ability to explicitly capture knowledge in the form of IF-THEN rules which are easily understandable by humans.

**Decision Tree Classifiers**

Decision tree classifiers such as ID3-type algorithms [22] and CART [6] are a powerful tool in multistage, hierarchical decision making. The basic approach here is to divide a complex decision-making process into a number of simpler and smaller problems [24]. The process of induction and structuring of a decision tree essentially begins by collecting a set of past examples which best describe instances of an object or an event that is to be classified. In the case of the previously-mentioned ultrasonic flaw detection expert system, digitized signals from known flawed and unflawed sources are first correctly classified by the expert and then provided to the learning algorithm. This set which is called the *training set* captures salient features of each signal along several dimensions in conjunction with the correct classification label. For instance, some of the features selected in our example are mean, variance, skewness and kurtosis. Each training example then associates with a specific set of feature-value pairs a predetermined class label. This point is illustrated in Table 1.

At the outset of tree induction, a null decision tree is formed which consists of an empty root and the entire set of training examples. At this level of tree construction, the root node has to be split by generating a test based on a feature and a feature value. For example, given the ten training examples shown in Table 1, a partial tree (see Fig. 1) is generated by selecting kurtosis-0.19 as the most salient feature-value pair. The test based on kurtosis-0.19 expands the root node into left and right branches such that the training examples in Table 1
Table 1. The training set

<table>
<thead>
<tr>
<th>Training Example</th>
<th>Mean</th>
<th>Variance</th>
<th>Skewness</th>
<th>Kurtosis</th>
<th>Signal Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.62</td>
<td>0.52</td>
<td>0.63</td>
<td>0.64</td>
<td>Flaw</td>
</tr>
<tr>
<td>2</td>
<td>0.38</td>
<td>0.21</td>
<td>0.61</td>
<td>0.31</td>
<td>Noise</td>
</tr>
<tr>
<td>3</td>
<td>0.61</td>
<td>0.48</td>
<td>0.50</td>
<td>0.54</td>
<td>Flaw</td>
</tr>
<tr>
<td>4</td>
<td>0.14</td>
<td>0.87</td>
<td>0.60</td>
<td>0.26</td>
<td>Flaw</td>
</tr>
<tr>
<td>5</td>
<td>0.19</td>
<td>0.10</td>
<td>0.36</td>
<td>0.37</td>
<td>Noise</td>
</tr>
<tr>
<td>6</td>
<td>0.42</td>
<td>0.25</td>
<td>0.46</td>
<td>0.14</td>
<td>Noise</td>
</tr>
<tr>
<td>7</td>
<td>0.21</td>
<td>0.11</td>
<td>0.27</td>
<td>0.55</td>
<td>Noise</td>
</tr>
<tr>
<td>8</td>
<td>0.58</td>
<td>0.52</td>
<td>0.47</td>
<td>0.70</td>
<td>Flaw</td>
</tr>
<tr>
<td>9</td>
<td>0.64</td>
<td>0.41</td>
<td>0.30</td>
<td>0.02</td>
<td>Noise</td>
</tr>
<tr>
<td>10</td>
<td>0.13</td>
<td>0.08</td>
<td>0.66</td>
<td>0.93</td>
<td>Noise</td>
</tr>
</tbody>
</table>

whose kurtosis components have values less than 0.19 fall in the left branch (examples 6 and 9) while the remaining cases follow the right branch. Mention must be made that determination of a feature-value pair at a given node is not an arbitrary procedure. Generally, partitioning of a tree node first requires examination of all possible feature-value pairs and then selection of a test which maximizes some fitness criterion such as minimum entropy, minimum error or some other heuristics based on statistical significance [20].

< 0.19           >= 0.19

Fig. 1. An induced tree

Once a node is split into two branches, the above-mentioned tree construction procedure can be recursively applied to all the newly generated nodes until some stopping criterion is met. For instance, if all training examples in a node belong to a specific class (e.g. Flaw), then expansion of the tree along that particular branch stops and the terminal node is marked
as a *leaf*. Otherwise, tree construction proceeds as before. To complete our example, a decision tree that classifies signals as *Flaw* or *Noise* based on the training information in Table 1 is shown in Fig. 2.

The decision tree shown in Fig. 2 can in fact be converted into *IF-THEN* rules by interpreting all the node tests along a particular tree branch as conjunction of several logical tests:

![Decision Tree Diagram](image)

**Rule 1:** IF (Kurtosis < 0.19)  
THEN Class = Noise

**Rule 2:** IF [(Kurtosis >= 0.19) AND (Variance >= 0.23)]  
THEN Class = Flaw

**Rule 3:** IF [(Kurtosis >= 0.19) AND (Variance < 0.23)]  
THEN Class = Noise
Furthermore, once a decision tree is generated, the problem of classification of future cases whose classification labels are missing simply reduces to traversing the tree starting from the root, performing appropriate node tests, and following branches until a leaf node is reached which indicates Flaw or Noise.

Some relevant issues warrant further attention at this point. First, in many real-world applications due to a number of reasons training and testing information may become corrupt. For example, feature-value measurements may be tainted by noise or become unknown or missing. And second, availability of new training information over time may require updating of the current knowledge base. Therefore, in order to maintain the performance of a classifying system at acceptable levels it is imperative to devise mechanisms to deal with these anomalous situations.

First, in terms of effects of presence of noise in the training information, construction of a tree can sometimes lead to the phenomenon of overfitting which occurs when a decision tree produced in a hierarchical manner often becomes more complex than can be justified by the given training data. In other words, the tree building mechanism attempts to create subtrees which fit the noise that exists in the training information. To circumvent this problem, tree pruning techniques such as pessimistic error pruning [19,20] or cost complexity pruning [6] have been devised which usually result in smaller trees and have the additional advantage of increased accuracy when classifying unseen objects.

The second issue, which can also become the bottleneck of many learning systems, is presence of missing and/or unknown information in the training/testing data regarding one or more feature-value pairs. For example, in the case of our flaw detection expert system, it is possible that a few training examples are presented to the learning system as presented in Table 2.
Table 2. Training examples with missing feature values

<table>
<thead>
<tr>
<th>Training Example</th>
<th>Mean</th>
<th>Variance</th>
<th>Skewness</th>
<th>Kurtosis</th>
<th>Signal Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>unknown</td>
<td>unknown</td>
<td>0.63</td>
<td>0.64</td>
<td>Flaw</td>
</tr>
<tr>
<td>12</td>
<td>0.38</td>
<td>0.21</td>
<td>missing</td>
<td>0.31</td>
<td>Noise</td>
</tr>
</tbody>
</table>

In cases such as above, therefore, CART’s surrogate splits and ID3’s information theoretic technique among several others, have been proposed in order to circumvent partial training/testing information during the learning process [21].

And finally, the issue of updating the current knowledge base is also important in real-world applications when new training information arrives over time. In the case of availability of new materials information, for instance, a decision tree classifier must be able to refine old concepts or perhaps form new ones. Traditional expert systems handle this knowledge updating through manual intervention, i.e., experts have to explicitly hardwire new concepts into the knowledge base. Incremental ID3-type decision tree learners such as IDea [12], on the other hand, have the capability to automatically incorporate the new training information into the current knowledge base with minimal human intervention.

Materials Property Databases

Selection of materials in design and manufacturing is a difficult and laborious task which was traditionally performed by means of reference books, product data sheets or catalogs. With the rapid pace of engineering design, however, manufacturing industries have realized the need for developing computerized systems which can expediently and precisely capture and communicate materials information for various applications [15].

In general, computerized materials databases are of two types: proprietary and commercial. Proprietary databases are primarily designed by various organizations in response to
their specific needs. For example, the Honeywell Space Inertial Guidance & Navigational Systems commissioned the construction of a database containing 6000 nonferrous isotropic materials entries specifically intended for the design of avionics [16]. Commercially available systems on the other hand tend to be concerned with larger and more general purpose classes of materials. These include: Rapra Technology's PLASCAMS system designed for polymers, Engineering Information Company's MATUS class of databases for metals, NIST's MTDATA for ceramics, and finally, in terms of multi-material databases, PDA Engineering's M/VISION and ASM International Mat.DB databases [18].

Mat.DB, which was particularly selected for our applications needs, is a database management system designed for maintaining information on the properties and processing of a variety of engineered materials such as aluminum, copper, magnesium, stainless steel, alloy steel, titanium, structural steel and plastics [1]. The basic building block in Mat.DB is the material record which contains information on a given material including designations, specifications, composition, properties at various operating temperatures, classes such as heat resistant, high strength, etc., and finally, processing characteristics.

One of the powerful features of Mat.DB is its ability to search through all available fields to locate needed material data. For example, users of the database can perform range searching on compositions or simply search for property values at specified temperatures. More importantly, in order to discover qualitative and quantitative relationships between various materials properties and performance measures, Mat.DB offers a utility program by which the entire contents of a specific material database can be converted into a plain text file. This utility is of paramount importance in machine learning applications where a learning algorithm requires a set of training examples in order to explicate classification knowledge regarding a specific task at hand. This point will be further clarified in the next section.
Classification of Performance Criteria

In this section we present the methodology for discovering the knowledge which actually determines how mechanical properties such as tensile elongation, tensile yield strength, hardness, etc. determine a material's performance measures such as formability, weldability, wear resistance, and stress corrosion resistance. In general, classification of materials in Mat.DB is accomplished by an indexing system which allows access to the database through two types of performance criteria: *application classes* and *rankings*. Application classes for metals such as stainless steel and alloy steel, for example, include classification labels such as fatigue resistant, wear resistant, creep resistant, high strength and stress corrosion resistant. Rankings, on the other hand, allow searching the database by processing characteristics such as formability, machinability and weldability. In contrast to application classes which are binary concepts, i.e., they are either present or not present, rankings are multicategorical and their characteristics are designated by letters A through F. For instance, explanation of letter designations used to denote various degrees of formability for specific metals is depicted in Table 3.

<table>
<thead>
<tr>
<th>Designation</th>
<th>Formability Characteristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>High: can be drawn into deep cuts at room temperature without use of excessive amounts of power or special lubricants</td>
</tr>
<tr>
<td>B</td>
<td>Good: can be drawn into deep cuts at room temperature but requires more power</td>
</tr>
<tr>
<td>C</td>
<td>Fair: can be drawn into relatively intricate shapes but requires heating of the blanks and powerful presses</td>
</tr>
<tr>
<td>D</td>
<td>Difficult: very difficult to cold form and susceptible to cracking</td>
</tr>
<tr>
<td>E</td>
<td>Poor: some forming can be done by either heating the blanks or process annealing between forming operations</td>
</tr>
<tr>
<td>F</td>
<td>Very poor: any amount of cold forming is virtually impossible without danger of cracking</td>
</tr>
</tbody>
</table>
In order to learn the aforementioned performance criteria, it was first necessary to con­vert the relevant information in the Mat.DB database into a suitable form recognizable by the decision tree classifier. As we mentioned before, an ID3-type classifier requires a set of training examples where each example, given an object's essential properties or features, correctly classifies that object. In our application, therefore, a training example contains sali­ent mechanical properties of a material along with appropriate performance criteria.

In terms of mechanical characteristics by which a material is classified, Mat.DB con­tains approximately 60 properties for various types of metals. The stainless steel and alloy steel databases, for example, which were the only two metals databases containing any infor­mation regarding the two types of performance measures, included a total of 20 mechanical properties out of which only five properties were deemed as most crucial based on our analysis. These include: tensile elongation, tensile ultimate strength, tensile yield strength, tensile reduction in area, and finally, hardness. Table 4 presents these features along with their proper abbreviations.

<table>
<thead>
<tr>
<th>Material Property (feature)</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tensile elongation</td>
<td>TenElong</td>
</tr>
<tr>
<td>Tensile ultimate strength</td>
<td>TenUltStr</td>
</tr>
<tr>
<td>Tensile yield strength</td>
<td>TenYldStr</td>
</tr>
<tr>
<td>Tensile reduction in area</td>
<td>TenRdArea</td>
</tr>
<tr>
<td>Hardness</td>
<td>Hardness</td>
</tr>
</tbody>
</table>

Once the set of materials properties was determined, the next stage in preparation of training examples was to retrieve relevant data from Mat.DB and to convert them into the proper format. To this end, we used a Mat.DB utility program which could download any particular material database in its entirety into a plain text file. Naturally, the generated text
file contained a variety of information such as designations, specifications, graphs and forms in addition to mechanical properties and various performance criteria. Therefore, it was necessary to filter out irrelevant information from this file and retain only the five selected mechanical properties listed in Table 4 from each database record along with the selected binary and multicategorical performance measures listed in Tables 5 and 6, respectively.

<table>
<thead>
<tr>
<th>Application Classes (Categories: Present, Not Present)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creep Resistant</td>
</tr>
<tr>
<td>Fatigue Resistant</td>
</tr>
<tr>
<td>High Strength</td>
</tr>
<tr>
<td>High Toughness</td>
</tr>
<tr>
<td>Stress Corrosion Resistant</td>
</tr>
<tr>
<td>Wear Resistant</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Processing Characteristics (Categories: A, B, C, D, E, F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formability</td>
</tr>
<tr>
<td>Machinability</td>
</tr>
<tr>
<td>Weldability</td>
</tr>
</tbody>
</table>

The Mat.DB utility program was able to produce a plain text file containing a total of 1062 records relating to the stainless steel database which was initially selected as the material of choice in the learning experiment. Our developed filtering program in turn produced two sets of training examples. The first set, including a total of 81 training examples, retained material properties for stainless steel at the temperature of 20 degrees C along with classification labels pertaining to binary application classes listed in Table 5. The second set of training examples which consisted of 84 training examples included the same set of material properties and the multicategorical performance measures presented in Table 6. Henceforth, the aim of the learning task in this set of experiments was twofold: first, to discover how material properties can be used to classify stainless steel by application classes, and sec-
ond, to ascertain how processing characteristics of stainless steel materials are affected by the same set of mechanical properties.

It must be emphasized that, in general, number of training examples plays a crucial role in learning in that too few examples may not be able to endow a learning algorithm with the necessary generalization power [27]. To compensate for the small number of training examples extracted from stainless steel database, therefore, we devised a second set of experiments whereby the above-mentioned process was completely repeated for the stainless steel and alloy steel databases used in conjunction. Table 7 summarizes these findings.

Table 7. Number of training examples extracted for performing learning

<table>
<thead>
<tr>
<th>Materials Database</th>
<th>Number of Plain Text Records Retrieved by Mat.DB Utility Program</th>
<th>Number of Training Examples Extracted for Learning the Application Classes Concepts (Table 5)</th>
<th>Number of Training Examples Extracted for Learning the Processing Characteristics Concepts (Table 6)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stainless steel</td>
<td>1062</td>
<td>81</td>
<td>84</td>
</tr>
<tr>
<td>Stainless steel and alloy steel</td>
<td>2300</td>
<td>180</td>
<td>208</td>
</tr>
</tbody>
</table>

Subsequently, the results presented in the next two sections pertain to two different sets of experiments. In the first set of experiments, the decision tree learning algorithm relies on the training information extracted only from the stainless steel database, while in the second set, the set of training examples generated from stainless steel is augmented with that of alloy steel. Moreover, the results of each set of experiments are further divided into two parts. The first part of each experiment relates to discovery of knowledge which governs the nature of relationship between material properties and application classes concepts such as fatigue.
resistant, creep resistant, wear resistant, etc. Second parts of the two experiments, on the other hand, concentrate on extraction of classification knowledge about the processing characteristics of materials which include formability, machinability and weldability.

Experiment 1: Learning Application Classes

In this experiment, two sets of training examples were retrieved from Mat.DB: 81 examples from the stainless steel database alone and 180 examples from both the stainless steel and alloy steel databases. For each type of material, therefore, a training example was formed which contained values of various mechanical properties listed in Table 4 in conjunction with material’s appropriate application class. An interesting point is that information regarding mechanical properties was quite sparse in the stainless steel database (see Table 8). Using the combination of stainless steel and alloy steel databases, however, drastically reduced the percentages of the missing material’s properties information. Furthermore, the presence of missing feature values was dealt with probabilistically in the manner described in the section on decision tree classifiers.

<table>
<thead>
<tr>
<th>Material Property (Feature)</th>
<th>Missing Data in Stainless Steel</th>
<th>Missing Data in Stainless Steel and Alloy Steel</th>
</tr>
</thead>
<tbody>
<tr>
<td>TenElong</td>
<td>28%</td>
<td>16%</td>
</tr>
<tr>
<td>TenUltStr</td>
<td>25%</td>
<td>14%</td>
</tr>
<tr>
<td>TenYldStr</td>
<td>26%</td>
<td>15%</td>
</tr>
<tr>
<td>TenRdArea</td>
<td>60%</td>
<td>31%</td>
</tr>
<tr>
<td>Hardness</td>
<td>74%</td>
<td>36%</td>
</tr>
</tbody>
</table>
The frequency distribution of various application classes with binary categories is also reported in Table 9. Representation of binary classification labels for the six application class categories depicted in Table 5 was achieved by two types of letter designations: \( P \) (concept present) and \( NP \) (concept not present). For example, in the case of the creep resistant concept in Table 9, out of 81 training examples, 41 types of stainless steel material at 20 degrees Celsius were classified as being creep resistant (marked \( P \)) while the remaining 40 cases did not manifest any such behavior (marked \( NP \)).

<table>
<thead>
<tr>
<th>Concept</th>
<th>Stainless Steel</th>
<th>Stainless Steel and Alloy Steel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( P )</td>
<td>( NP )</td>
</tr>
<tr>
<td>Creep Resistant</td>
<td>41</td>
<td>40</td>
</tr>
<tr>
<td>Fatigue Resistant</td>
<td>44</td>
<td>37</td>
</tr>
<tr>
<td>High Strength</td>
<td>45</td>
<td>36</td>
</tr>
<tr>
<td>High Toughness</td>
<td>80</td>
<td>1</td>
</tr>
<tr>
<td>Stress Corrosion Resistant</td>
<td>56</td>
<td>25</td>
</tr>
<tr>
<td>Wear Resistant</td>
<td>4</td>
<td>77</td>
</tr>
</tbody>
</table>

During the next stage of learning, six decision tree classifiers were trained on each of the two available sets of training examples: one extracted from the stainless steel database with 81 examples and the other pertaining to both stainless steel and alloy steel materials containing 180 examples. After completion of the training phase in each case, the generated decision trees had to be tested in order to determine the quality of the extracted knowledge. In general, classifier systems rely on the widely-used resampling technique of train-and-test where a set of training examples is divided into two portions: a training set and a testing set. The training set is used in its entirety to train a learning system while the testing set is used.
to obtain an unbiased estimate of the true performance of the trained learning system [27]. In our case, however, due to shortage of training information, it was decided to train and test a decision tree on the same set of training examples and obtain the apparent error rates which are obviously more optimistic measures of an induced tree’s true gauge of performance.

Furthermore, the accuracy measures listed in Table 10 are divided into two categories: POD and POF. The POD measure is an indicator of the percent of examples classified as having a performance parameter present providing that the performance parameter is present, so higher POD values imply more accurate results. The POF measure, on the other hand, signifies the fraction of examples classified as having a performance parameter present given that the performance parameter is not present, and hence, lower POF values indicate higher quality classification knowledge. To illustrate, the POD measure of 100% in the first row of Table 10 indicates that the decision tree classifier can correctly classify all creep resistant stainless steel materials. The POF measure of 12%, however, shows that the classifier misclassified 12.5% of the cases by attributing creep resistance to some materials which do not exhibit such characteristic.

Table 10. Accuracy of the trained decision trees in learning application class concepts

<table>
<thead>
<tr>
<th>Concept</th>
<th>Stainless Steel</th>
<th>Stainless Steel + Alloy Steel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy Measures (Category P)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>POD</td>
<td>POF</td>
</tr>
<tr>
<td>Creep Resistant</td>
<td>100%</td>
<td>12%</td>
</tr>
<tr>
<td>Fatigue Resistant</td>
<td>82%</td>
<td>0%</td>
</tr>
<tr>
<td>High Strength</td>
<td>84%</td>
<td>0%</td>
</tr>
<tr>
<td>High Toughness</td>
<td>96%</td>
<td>0%</td>
</tr>
<tr>
<td>Stress Corrosion Resistant</td>
<td>75%</td>
<td>28%</td>
</tr>
<tr>
<td>Wear Resistant</td>
<td>50%</td>
<td>0%</td>
</tr>
</tbody>
</table>
Results presented in Table 10 also demonstrate two important facts. First, the decision tree algorithm is robust in that although the percentages of missing materials properties were quite high in some instances (see Table 8), the quality of the extracted classification knowledge was still above acceptable levels in most instances. And second, the quality of extracted classification knowledge improved in some instances by augmenting the stainless steel training examples with those of alloy steel cases. For example, the (POD, POF) measure of (50%, 0%) for stainless steel material improved to (67%, 2%) while using the two databases in conjunction. However, (96%, 0%) accuracy of learning the high toughness performance criterion for stainless steel material degraded to (48%, 0%) when the two types of steel materials were combined. This degradation in performance can be directly attributed to two factors. First, the highly skewed distribution of the high toughness concept in both sets of training examples (see Table 9) implies that a classifier does not have enough information relating to presence and lack of presence of the high toughness concept. This could, consequently, severely impair a classifier’s ability to draw valid generalizations. And second, the combination of the two types of steel materials with their own distinct mechanical properties could perhaps have introduced some anomalies regarding accurate classification of the high toughness concept.

Finally, in the case of each of the six application classes, an induced decision tree was able to identify the relationship between a material’s mechanical properties and its appropriate type of classification. For the sake of brevity, however, we only show the results on the fatigue resistant concept for stainless steel material which was acquired with the (POD, POF) accuracy measure of (82%, 0%) on the 81 training examples:

\[ IF \left( TenElong < 25\% \right) \ \& \ \left( Hardness < 34.5 \text{ H} \right) \ \ OR \]
\[ (\text{TenElong} < 25\%) \text{ AND } (\text{Hardness} \geq 34.5\text{H}) \text{ AND } (\text{TenUltStr} \geq 1422.5 \text{ MPa}) \text{ OR } \]
\[ 44\% \leq \text{TenElong} < 47\% \]

THEN \hspace{1cm} \text{Material is Fatigue Resistant}

Experiment 2: Learning Processing Characteristics

To acquire the processing characteristics concepts such as formability, machinability and weldability, two sets of training examples were used in the learning experiments: one set containing 84 examples which related to the stainless steel material, and a second set with 208 examples from both the stainless steel and alloy steel databases. Fractions of the missing material properties used in the training sets, and also, the frequency distribution of the three classification concepts are shown in Tables 11 and 12, respectively.

The frequency distributions of concepts as shown in Tables 12a and 12b again indicate some skewness in representation of some of the concept categories. For instance, to learn high formability or high machinability for stainless steel material (category A), the decision tree algorithm had only one example out of 84 to draw the necessary generalizations. Categories B through F for each of the three processing characteristics, however, were more evenly distributed. Also, due to different nature of processing characteristics and properties of stainless and alloy steel materials, it was further decided to transform the six-category representations shown in Table 12b into a finer classification problem with only two categories. To this end, the class labels A through C in Table 12b were collapsed into a single new category A and labels D through F were collapsed into B. Table 13 shows the new frequency distributions of the processing characteristic concepts used for the stainless and alloy steel material databases.
The POD and POF accuracy measures of the induced classifiers for each of the processing characteristics are shown in Tables 14a and 14b. In the case of stainless steel material, acquisition of concepts such as weldability proved to be a difficult task considering the spare nature of information available in the database. Nevertheless, high formability, high machinability and high weldability concepts (category A) were learned at acceptable rates.

Table 11. Percentages of the missing properties

<table>
<thead>
<tr>
<th>Material Property (feature)</th>
<th>Stainless Steel Missing Data</th>
<th>Stainless Steel + Alloy Steel Missing Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>TenElong</td>
<td>30%</td>
<td>16%</td>
</tr>
<tr>
<td>TenUltStr</td>
<td>26%</td>
<td>14%</td>
</tr>
<tr>
<td>TenYldStr</td>
<td>27%</td>
<td>15%</td>
</tr>
<tr>
<td>TenRdArea</td>
<td>61%</td>
<td>32%</td>
</tr>
<tr>
<td>Hardness</td>
<td>75%</td>
<td>37%</td>
</tr>
</tbody>
</table>

Table 12. Frequency distributions for processing characteristics concepts

(a) Stainless steel

<table>
<thead>
<tr>
<th>Concept</th>
<th>Category Label</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formability</td>
<td></td>
<td>1</td>
<td>13</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>34</td>
</tr>
<tr>
<td>Machinability</td>
<td></td>
<td>1</td>
<td>3</td>
<td>13</td>
<td>29</td>
<td>22</td>
<td>16</td>
</tr>
<tr>
<td>Weldability</td>
<td></td>
<td>4</td>
<td>19</td>
<td>10</td>
<td>4</td>
<td>28</td>
<td>19</td>
</tr>
</tbody>
</table>

(b) Stainless steel and alloy steel

<table>
<thead>
<tr>
<th>Concept</th>
<th>Category Label</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formability</td>
<td></td>
<td>1</td>
<td>14</td>
<td>20</td>
<td>25</td>
<td>50</td>
<td>98</td>
</tr>
<tr>
<td>Machinability</td>
<td></td>
<td>1</td>
<td>5</td>
<td>46</td>
<td>69</td>
<td>50</td>
<td>37</td>
</tr>
<tr>
<td>Weldability</td>
<td></td>
<td>4</td>
<td>22</td>
<td>21</td>
<td>24</td>
<td>67</td>
<td>70</td>
</tr>
</tbody>
</table>
Table 13. New frequency distributions for processing characteristics concepts

<table>
<thead>
<tr>
<th>Concept</th>
<th>Stainless Steel + Alloy Steel</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Category A</td>
</tr>
<tr>
<td>Formability</td>
<td>35</td>
</tr>
<tr>
<td>Machinability</td>
<td>52</td>
</tr>
<tr>
<td>Weldability</td>
<td>47</td>
</tr>
</tbody>
</table>

Table 14. Accuracy of the decision tree on learning the three processing characteristics concepts

(a) Stainless steel

<table>
<thead>
<tr>
<th>Category</th>
<th>Formability</th>
<th>Machinability</th>
<th>Weldability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>POD</td>
<td>POF</td>
<td>POD</td>
</tr>
<tr>
<td>A</td>
<td>100%</td>
<td>0%</td>
<td>100%</td>
</tr>
<tr>
<td>B</td>
<td>69%</td>
<td>6%</td>
<td>100%</td>
</tr>
<tr>
<td>C</td>
<td>55%</td>
<td>4%</td>
<td>31%</td>
</tr>
<tr>
<td>D</td>
<td>50%</td>
<td>0%</td>
<td>59%</td>
</tr>
<tr>
<td>E</td>
<td>54%</td>
<td>8%</td>
<td>23%</td>
</tr>
<tr>
<td>F</td>
<td>85%</td>
<td>6%</td>
<td>50%</td>
</tr>
</tbody>
</table>

(b) Stainless steel and alloy steel

<table>
<thead>
<tr>
<th>Category</th>
<th>Formability</th>
<th>Machinability</th>
<th>Weldability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>POD</td>
<td>POF</td>
<td>POD</td>
</tr>
<tr>
<td>A</td>
<td>80%</td>
<td>12%</td>
<td>75%</td>
</tr>
<tr>
<td>B</td>
<td>88%</td>
<td>20%</td>
<td>85%</td>
</tr>
</tbody>
</table>
Moreover, in the case of the combined stainless and alloy steel databases (see Table 14b), the generated decision trees were able to classify concepts of interest with relatively high accuracy. For example, trained on 208 training examples, a decision tree was able to classify fair to high grades of formability (new category A or old categories A and B and C) with (POD, POF) measure of (80%, 12%) which is quite acceptable.

Finally, in closing this section, we list three discovered rules which describe the relationship between mechanical properties of stainless steel material and each of the three processing characteristics of formability, machinability and formability.

**Rule 1:**

IF \[ (372.0 \text{ MPa} \leq \text{YldStr} < 695.5 \text{ MPa}) \text{ AND } \left\{ \text{UltStr} \geq 961.5 \right\} \]

THEN

Formability is high

i.e., material can be drawn into deep cuts at room temperature without use of excessive amounts of power or special lubricants

**Rule 2:**

IF \[ (\text{YldStr} < 548.0 \text{ MPa}) \text{ AND } (\text{Elong} < 25\%) \]

THEN

Machinability is high

i.e., Material can be easily machined by any or all types of operations at maximum feeds, speeds and depths of cut. Selection of cutting fluid is not critical and cutting fluid is often not required.
Rule 3:

IF [(TenUltStr < 474.0 MPa)]

THEN

Weldability is high

i.e., material is readily weldable by virtually all methods without the necessity for preheating or postheating.

Conclusions

With the rapid pace of growth in engineering design and manufacturing, it is crucial for designers, who may not be expert process or analysis engineers, to have a working knowledge of materials selection and performance issues. In absence of real materials experts, however, searching for various types of materials information is a time-consuming task which requires extensive examination of product sheets, catalogs, reference books, or property databases.

In this paper, therefore, we introduce a new technique for assisting designers in selecting appropriate materials. Specifically, we use an ID3-type decision tree learning system that can automatically discover knowledge of materials selection from a commercially available property database. This knowledge is represented in the familiar form of IF-THEN rules which are readily made available to designers. Moreover, our developed ID3-type classifier is suitable for real world-applications where training materials information may be imprecise/missing or become available incrementally over time.
References


Abstract

In this paper we introduce a methodology within which multiobjective design optimization is approached from an entirely new perspective. Specifically, we demonstrate that multiple-objective optimization through induction of multivariate regression trees is a powerful alternative to the conventional vector optimization techniques. Furthermore, in an attempt to investigate the effect of various types of splitting rules on the overall performance of the optimizing system, we present a tree partitioning algorithm which utilizes a number of techniques derived from diverse fields of statistics and fuzzy logic. These include: two multivariate statistical approaches based on dispersion matrices, an information-theoretic measure of covariance complexity which is typically used for obtaining multivariate linear models, two newly-formulated fuzzy splitting rules based on Pearson’s parametric and Kendall’s nonparametric measures of association, Bellman and Zadeh’s fuzzy decision-maximizing approach within an inductive framework, and finally, the multidimensional extension of a widely-used fuzzy entropy measure.

In terms of potential application areas, we highlight the advantages of our methodology by presenting the problem of multiobjective design optimization of a beam structure.
Introduction

Traditionally, tree-structured approaches to regression such as classification and regression trees (CART) [4] and inductive partitioning with regression trees (IPRT) [19] algorithms have been developed to mainly perform exploratory data analysis involving multiple independent variables and single responses or outcomes. These algorithms are powerful in that not only do they perform ordinary regression, but they also learn regression surfaces by extracting the knowledge that governs the input-output behavior of the model under consideration in the form of regression trees. This form of knowledge acquisition is of paramount importance when regression trees are used for performing optimization tasks. The case in point is that an induced tree essentially represents complex regression surfaces in terms of a number of simpler regression subsurfaces. Detailed examination of these subsurfaces therefore can potentially identify design regions where a product or process response is optimized. Consequently, in addition to pinpointing optimal response regions, tree-structured approaches to optimization offer the advantage of explicating the knowledge that actually constitutes the optimality of the generated solutions.

In many real-world applications, however, the optimization problem at hand consists of several design variables which in turn specify the behavior of a number of responses. For example, in a quality control application, the main objective may be to discover the underlying knowledge which controls the performance of an electric discharge machining (EDM) process in which process variables such as pulse duration and discharge current directly determine several process responses such as electrode wear, surface roughness and metal removal rate [15]. Evidently, in these situations approaches such as CART and IPRT are insufficient due to their single-response limitations. Conventional methods such as response surface methodology [21], vector optimization techniques [6] and statistical multivariate
analysis of variance (MANOVA) [11], on the other hand, are also limiting in that they fail to provide an understanding of the complex interrelationships which exist between various components of the process which is to be optimized. Based on this observation, the artificial intelligence community has approached this problem by placing special emphasis on development of symbolic search techniques. For example, the MOA* algorithm, although limited in its applicability to real-world problems, was developed as a multiobjective generalization of the heuristic search algorithm A* [3].

Another important issue relating to multivariate regression trees which also has not been investigated previously is the choice of splitting rules for successive partitioning of a tree during the induction process. Essentially, starting with an empty root node, a tree is recursively grown by partitioning its terminal leaves until some stopping criterion is met. In the case of univariate regression analysis, the CART and IPRT algorithms rely exclusively on the traditional statistical concepts. It must be emphasized, however, that similar to decision tree algorithms, the choice of partitioning criteria in tree-structured approaches to regression is crucial in that it directly determines the quality of the extracted knowledge [5].

To summarize, in this paper we present a new framework within which multiobjective optimization is accomplished through induction of multivariate regression trees. Furthermore, we present a tree partitioning algorithm which utilizes a number of splitting rules based on concepts from statistics and fuzzy logic. Obviously, the choice of using the traditional statistical formulations in this work was instigated by the historic fact that statistics is a firmly established science with many facets which render it a particularly viable tool in many scientific applications. The theory of fuzzy sets [9], on the other hand, is a more recently developed concept, and it too has proven to be an invaluable tool in a wide array of applications ranging from pattern recognition and clustering to design of digital circuits and relational data bases [12, 16]. In fact, within the context of multiobjective optimization,
Bellman and Zadeh's fuzzy approach to optimization [1] has been widely implemented in many engineering structural optimization applications [17]. Therefore, in an attempt to examine the effect of various types of tree partitioning rules on the overall learning process, and also, to assess the feasibility of techniques based on fuzzy logic we describe seven splitting rules. Specifically, these include: two statistical decision rules based on dispersion matrices, a statistical measure of covariance complexity which is typically used for obtaining multivariate linear models [2], two newly-formulated fuzzy partitioning methods based on Pearson's parametric [11] and Kendall's nonparametric [20] measures of association, Bellman-Zadeh's decision-maximizing fuzzy approach [1] to optimization in an inductive framework, and finally, the multidimensional extension of a widely-used measure of fuzzy entropy [13].

The remainder of this paper is organized as follows. In order to make the paper self-contained, Section 2 briefly reviews the theory of fuzzy sets. Section 3 describes our methodology for transforming the problem of multiobjective optimization into induction of multivariate regression trees using fuzzy and nonfuzzy splitting criteria. Section 4 presents key results of applying techniques described in this paper to a multiobjective design problem. And finally, Section 5 summarizes the paper.

Fuzzy Logic

Since its introduction in the 1960's, the theory of fuzzy sets has matured to such an extent that any detailed examination of its various components will be beyond the scope of this paper. In this section therefore we introduce some fundamental definitions of fuzzy set theory in order to make the next section's presentation self-contained. The interested reader however may wish to refer to the multitude of sources available on this topic to date including Kandel's [12] excellent treatment of the subject.
The theory of fuzzy sets deals with a subset $A$ of the universe of discourse $X$, where the transition between full membership and no membership is gradual rather than abrupt. Traditionally, the grade of membership 1 is assigned to objects that fully belong to $A$, while 0 is assigned to objects that do not belong to $A$ at all. In other words, the more an object $x$ belongs to $A$, the closer to 1 its grade of membership $\mu_A(x)$ [12].

The assignment of the membership function of a fuzzy set is a crucial step since it determines how an individual member’s degree of belongingness should vary between 0 and 1. For example, a membership function can exhibit linear or nonlinear behavior depending upon the subjective preferences of the user and the context of the problem at hand [18]. Furthermore, the shape of a membership function can also be controlled in order to incorporate various fuzzy preferences such as smaller-is-better, larger-is-better or nominal-is-the-best [16].

Before proceeding to the next section, it must be pointed out that similar to the ordinary set theory, fuzzy sets are manipulated by fuzzy interpretations of the traditional set-theoretic aggregate connectives such as the complement, intersection and union operators. For instance, let $A$ and $B$ be two fuzzy subsets of $X$ with their corresponding membership functions $\mu_A(x)$ and $\mu_B(x)$, respectively. Hence, for all $x$ in $X$:

1. $\mu_C(x) = \text{Maximum}(\mu_A(x), \mu_B(x))$, where $C$ is the union of $A$ and $B$
2. $\mu_C(x) = \text{Minimum}(\mu_A(x), \mu_B(x))$, where $C$ is the intersection of $A$ and $B$
3. $\mu_{A^c}(x) = 1 - \mu_A(x)$, where $A^c$ is the fuzzy complement of $A$
Multivariate Regression Trees

The basic element for inducing a multivariate regression tree is a set of training examples which provides a capsule view into the objective/constraint space. These examples essentially enable the learning algorithm to incrementally construct a complex regression surface from a number of simpler regression subsurfaces. This piecewise model construction is accomplished in a top-down fashion by successive partitioning of the training population at each level of the tree in an attempt to identify compact clusters in the response region. Examination of these clusters in turn can identify location of the Pareto-optimal solution where an objective can be further improved only by degrading one or more objectives [6]. The following provides more details regarding the tree induction process.

Basically, given a learning sample \( L = (X_1,Y_1), (X_2,Y_2),\ldots, (X_N,Y_N) \), the learning algorithm produces a prediction rule \( d: \mathbb{R}^n \rightarrow \mathbb{R}^p \) which is a mapping from the n-dimensional predictor or attribute space \( (X_i)'s \) to the p-dimensional response (objectives and constraints) region \( (Y_i)'s \). The learning sample, therefore, contains \( N \) examples where each example associates a p-dimensional response vector with an n-dimensional predictor vector. Initially, all \( N \) examples reside at the root of an empty tree. Following a divide-and-conquer approach, the root node is split into two left and right nodes such that \( n_1 \) of the original \( N \) examples fall in the left node and the remaining \( n_2 \) cases in the right node \( (N = n_1 + n_2) \). This splitting is facilitated by selection of an attribute and a threshold for partitioning the attribute’s range into two regions [8]. Among all possible attribute/threshold pairs, the pair that results in the 'best' split, where the resulting left and right nodes maximize some measure of fitness, is selected and the node is split accordingly. The process of partitioning is then recursively applied to all newly generated nodes until some stopping criterion is met. In our
case, a multivariate heuristic which dictates that the number of examples in a node has to be at least as large as the number of responses was used. Furthermore, after a tree is completely grown in the prescribed manner, some type of pruning will prove beneficial should the problem of overspecialization cause detrimental effects on overall efficiency of the learning system [4].

After the learning phase is complete, the induced tree contains a number of paths which start from the root and end in a terminal node or leaf. Each path therefore pinpoints a regression subsurface by the virtue of examples that are contained in its leaf. A leaf's set of examples can be viewed as a cluster in the response region which is characterized by its mean vector $\mu$ and covariance $\Sigma$. The goodness of these clusters is in turn determined by a variety of statistical and fuzzy partitioning techniques which are explained below. In ensuing discussions assume that the response matrix at a given node is $R$ (m by p matrix) which contains $m$ p-dimensional response vectors and that covariance of $R$ is $\Sigma$. For fuzzy splitting criteria further assume that $R$ is converted to the multidimensional fuzzy set $M$ (m by p matrix) by fuzzifying individual responses $r_{ij}$ in $R$ into $\mu_{ij}$ in $M$ using one of the following (user option):

$$\mu_{ij}^{\text{linear}} = \frac{(r_{ij} - r_{ij}^{\text{min}})}{(r_{ij}^{\text{max}} - r_{ij}^{\text{min}})} \quad i=1,\ldots,m \text{ and } j=1,\ldots,p$$

$$\mu_{ij}^{\text{nonlinear}} = a [1 - \exp(-b \mu_{ij}^{\text{linear}})] \quad a \text{ and } b \text{ are user-defined} \ [18]$$

It must be mentioned that for a given response $j$, $r_{ij}^{\text{min}}$ and $r_{ij}^{\text{max}}$ are found by scanning rows $(i=1,\ldots,m)$ of $R$, and they can be interchanged depending upon whether the goal is to maximize or minimize the given response $j$ in the fuzzy domain.
The first two nonfuzzy splitting criteria (Methods 1 & 2) used for tree induction are the trace and determinant of the covariance matrix which denote the sum of individual response variances and the generalized variance, respectively [7]. Minimization of trace, which totally ignores the interaction among responses, attempts to locate spherically-shaped response clusters where individual variances are minimal. On the other hand, minimization of the generalized variance, $|\Sigma|$, helps identify parallelotopes formed by response vectors which have minimal volume [22].

The third partitioning rule (Method 3) uses Bozdogan's information-theoretic covariance complexity measure which is typically used for selection and evaluation of multivariate models [2]. Essentially, the covariance complexity metric measures how the individual subcomponents of a model or a system interact with one another. In the case of multivariate regression trees, we use a tree as a representative of an underlying model that is to be captured through the induction process. At each level of partitioning, a given node's original population of responses, i.e. $R$, is divided into two subpopulations in such a way that the covariance complexity of the resulting subpopulations are minimal. The overall task hence is to evaluate the degree of interaction that exists between responses in $R$ and select partitions which result in minimal entropy or disorder. This can be accomplished by assigning the following covariance complexity measure to the covariance matrix of a population $R$:

\[
(6) \quad CC(\Sigma) = 0.5 \cdot p \cdot \log_2 [\text{trace}(\Sigma)/p] - 0.5 \cdot \log_2 |\Sigma|
\]

where $p$ is the number of responses (objectives and constraints). During the tree splitting process then a parent node is partitioned into two nodes such that the measures of covariance complexity of the newly generated nodes are minimal.
The next two partitioning criteria (Methods 4 and 5) are based on Pearson’s parametric [11] and Kendall’s nonparametric [20] measures of association \( \rho \) and \( \tau \), respectively. The main motivation here is to discover the degrees of relationship between two responses \( R \) and \( S \) which may involve linear or nonlinear components. It must be emphasized that Pearson’s \( \rho \) is particularly suitable for situations where responses exhibit linear relationship. However, in many situations, linear approximations may become extremely misleading when the relationships involve nonlinear components. To this end, Kendall developed the correlation measure \( \tau \) which is not based on any parametric assumptions and is more likely to discover monotonic behavior between responses.

More formally, given the data \((R_1, S_1), ..., (R_N, S_N)\), Pearson’s degree of linear relationship \( \rho \) between responses \( R \) and \( S \) is:

\[
\rho_{RS} = \frac{\sum_{i=1}^{N} [(R_i - \text{mean}(R))(S_i - \text{mean}(S))] \sigma_R \cdot \sigma_S}
\]

where \( \sigma_R \) and \( \sigma_S \) are the standard deviations of \( R \) and \( S \), respectively. Also, Kendall’s degree of monotonic relationship between \( R \) and \( S \) is:

\[
\tau_{RS} = \frac{2}{N(N-1)} \left( \sum_{i<j} \text{sign}(R_i - R_j) \cdot \text{sign}(S_i - S_j) \right)
\]

where the sign function takes values +1, 0 or -1 depending upon whether its argument is positive, zero or negative. For the sake of simplicity, the following passage generically refers to \( \rho \) and \( \tau \) as \( \chi \) since the forthcoming analysis is symmetric with respect to both of these measures.
The measure of association $\chi$ attempts to discover the relationship between any two given responses. For example, if $R$ and $S$ tend to grow in a similar direction, $\chi_{RS}$ approaches 1. Conversely, if $\chi_{RS}$ approaches -1, it is concluded that $R$ and $S$ grow in opposite directions. Furthermore, $\chi_{RS}$ values near 0 imply absence of any relationship (linear in the case of $\rho$ and monotonic in the case of $\tau$) between the two responses. Considering this, we can now incorporate elements from fuzzy logic as follows. Assume that a particular node's set $M$ contains fuzzified responses as explained previously. Now, regardless of whether any individual response is to be maximized or minimized, the chief goal in the fuzzy domain is to locate regions where fuzzy responses approach their maximum values. Hence, given $M$, we obtain the matrix of correlation coefficients $T$ (a $p \times p$ matrix) where each $\chi_{ij}$ for responses $i$ and $j$ ($i,j = 1, \ldots, p$) is computed using Pearson's or Kendall's measures ($\chi_{ii} = 1$, $\chi_{i > j} = \chi_{i < j}$). Note that since $T$ is symmetric, only its above-diagonal elements, $\chi_{i < j}$, are considered for further calculations. These $p(p-1)/2$ elements, which are pairwise measures of association between fuzzy responses in $M$, take values between -1 and 1. However, the desired clusters to be found are those for which as many of these correlation values approach 1 as possible which simply means that all or most of the responses are approaching their expected extrema in a given region. To accomplish this, $T$'s above-diagonal $\chi_{ij}$ correlation coefficients are fuzzified using either linear or exponential membership function transformations. The aspiration levels of -1 and 1 are used in the fuzzification process to indicate that correlation values of 1 are desirable to attain maximum degree of belongingness. The cluster under consideration is then assigned the degree of trend fitness ($TF$):

\begin{equation}
TF(M) = \min_{k} \{ \mu(\chi_{k}) \} \quad k = 1, \ldots, p(p-1)/2
\end{equation}

The overall objective, therefore, is to identify splits for which the produced clusters have maximal TF measures.
The sixth splitting criterion (Method 6) is based on Bellman and Zadeh's approach to multiobjective optimization [1]. To give a brief overview, consider making a decision $D$ which can be seen as a confluence of $n$ objectives and constraints denoted by responses $R_1, \ldots, R_n$. The optimal decision in the fuzzy domain then can simply be viewed as the intersection of fuzzy sets $\mu(R_1), \ldots, \mu(R_n)$ where each $\mu(R_i)$ is calculated using appropriate membership function transformations. More formally, the optimization task can be formulated as finding an optimum predictor vector $X^*$ for which the measure:

\begin{equation}
\mu_D(X^*) = \min_i \{\mu(R_i(X))\}
\end{equation}

is maximized. Typically, after proper transformation of the problem at hand into the fuzzy domain, $X^*$ is found using nonlinear programming [17]. In our framework, however, Bellman-Zadeh's approach is used for partitioning a node such that the measure:

\begin{equation}
BZ(M) = \max_i \min_j \{\mu_{ij}\} \quad i=1, \ldots, m \text{ and } j=1, \ldots, p
\end{equation}

is maximized for a particular multidimensional fuzzy set $M$ under consideration.

The last inductive partitioning technique (Method 7) to be discussed relies on fuzzy entropy [13]. Basically, given a fuzzy set $A$ with its complement $A^c$, fuzzy entropy of $A$:

\begin{equation}
FE(A) = \frac{C^0(A, A^c)}{C^u(A, A^c)}
\end{equation}

measures how fuzzy actually $A$ is, where $C^0$ and $C^u$ denote counts of overlap and underlap between $A$ and $A^c$, respectively. In a top-down inductive approach, the fuzzy entropy measure can be used to identify fuzzy clusters $M$ which exhibit minimal amount of fuzziness at
each partitioning level. The basic definition of entropy, however, has to be extended so that fuzziness of the multidimensional fuzzy set \( M \) can be calculated. To accomplish this, first, \( M \)'s complement, \( M^c \), is calculated where each \( \mu_{ij}^c \) in \( M^c \) is complement of \( \mu_{ij} \) in \( M \). Then, fuzzy sets \( I \) and \( U \) (both \( m \) by \( p \) matrices), which denote the intersection and union of \( M \) and \( M^c \), are calculated where elements \( i_{ij} \) in \( I \) and \( u_{ij} \) in \( U \) are \( \min(\mu_{ij}, \mu_{ij}^c) \) and \( \max(\mu_{ij}, \mu_{ij}^c) \), respectively. Consequently, we define the fuzzy entropy measure of a multidimensional fuzzy set \( M \) as:

\[
(13) \quad \text{FE}(M) = \frac{\max_i \min_j \{i_{ij}\}}{\max_i \min_j \{u_{ij}\}} \quad i=1,\ldots,m \text{ and } j=1,\ldots,p
\]

During the course of tree induction, then, the attribute/threshold pair for which the resulting clusters have minimal fuzzy entropy are selected and the node is split accordingly.

One final comment concerning fuzzy logic-based methods 4 through 7 needs to be made at this point. The standard \( \min \) and \( \max \) operators are strict or pessimistic in the sense that their aggregated outputs can never exceed the highest input or be lower than the lowest input, respectively. Fuzzy set theory, however, provides a host of more optimistic aggregation connectives for integrating membership functions. In order to investigate the effects of some of these connectives on overall learning efficiency of the system, it was decided to incorporate the option of using either standard \( \min/\max \) operators or, alternatively, Yager's operators [14]:

\[
(14) \quad \text{max}_{\text{Yager}}(\mu_1, \mu_2) = \min(1, (\mu_1^w + \mu_2^w)^{1/w})
\]
where parameter $w > 0$ can be varied for achieving various degrees of optimism. An interesting fact to be mentioned is that for $w \gg 0$, Yager's min/max connectives converge to their standard definitions.

Optimization of a Beam Structure

In this section we present the problem of optimum design of a beam structure which was previously studied by Osyczka [15]. The design of the beam shown in Fig. 1 involves minimization of the two objectives of beam volume and the static compliance of the beam.

Fig. 1. The Beam Structure
The design variables in this example are $X_1$ and $X_2$, which represent the length of the part 1 of the beam and the interior diameter of the beam, respectively. Furthermore, the beam should resist the maximum force $F_{\text{max}} = 12000 \text{ N}$ and the permissible bending stress of the beam material is $\sigma_g = 180 \text{ N/mm}$. The overall optimization task can therefore be formally stated as follows:

**Objectives:**

Minimize the beam volume (17) and static compliance under the force $F$ (18):

(17) $F_1(X) = 0.785 [X_1 (D_2^2 - X_2^2) + (L - X_1)(D_1^2 - X_1^2)]$

(18) $F_2(X) = 3.2 \times 10^{-5} [((D_2^4 - X_2^4)^{-1} - (D_1^4 - X_1^4)^{-1})X_1^3 + L^3(D_1^4 - X_2^4)^{-1}]$

**Subject to:**

(19) $\sigma_g = 9.78 \times 10^6 \cdot X_1 (4.09 \times 10^7 - X_2^4)^{-1} \leq 180 \text{ N/mm}$

(20) $10.0 \leq X_1 \leq 300.0$

(21) $40.0 \leq X_2 \leq 75.2$

In regards to preparation of learning and testing cases, it was decided to sample the design region $(X_1, X_2)$ in 900 distinct points between $(10, 40)$ and $(300, 75.2)$ in order to ensure that the response surfaces were adequately represented to the learning algorithm. Following a widely-used variance-stabilizing technique, the objective and constraint responses were transformed into log domain [10] which helped the overall learning efficiency for
parametric as well as nonparametric induction criteria. The 900 points were then randomly shuffled and divided into two sets of size 450 each, namely, $T_{450}$ and $L_{450}$. The set $T_{450}$ was dedicated entirely to testing purposes while $L_{450}$ was used for the learning process. Samples of size 100, 150 and 200 were then randomly drawn from the overall learning set $L_{450}$. The learning phase then proceeded by inducing a regression tree on each of the randomly drawn samples $L_{100}$, $L_{150}$ and $L_{200}$ for each of the fuzzy and nonfuzzy splitting methods. This entire process of random selection of training samples, learning and testing was repeated a total of five times for each tree-growing technique so that results could be represented with 95% confidence (t-distribution). It must be mentioned that for this particular example, empirical studies indicated samples of size less than 100 (i.e., 100 function evaluations) were not able to convey enough information to the learning algorithm. In other problems, however, smaller samples may provide the learning algorithm with the necessary generalization power.

After completion of the learning phase, relative regression errors which normally vary between 0.0 (perfect regression model) and 1.0 (poor model) were computed for each induced tree. Our error analysis is similar to CART's [4] except that it was extended for multivariate cases by substituting Mahalanobis distances [7] for ordinary Euclidean-based error distances in order to account for covariances that exist among responses. Table 1 summarizes the relative regression errors for tree-partitioning methods 1 through 7.

The performance measures shown in Table 1 reveal three important facts. First, the overall inductive generalization power, and consequently, the regression accuracy of all partitioning techniques generally improves as the size of training sets increases from 100 to 200. Second, the accuracy of regression surfaces obtained through the use of fuzzy splitting criteria matches, and in few instances surpasses, the accuracy of solutions generated by well-established statistical techniques. And finally, application of fuzzy splitting rules for tree induction offers a level of flexibility which ordinary multivariate methods lack. For instance
switching from linear to nonlinear membership functions or using Yager's min/max definitions instead of standard connective operators frequently helped improve the quality of the obtained regression models. To illustrate this point, consider fuzzy splitting method 4 and $L_{200}$. The best results for this method were attained by using a combination of linear membership functions and Yager's min/max aggregate connectives. For method 6 and $L_{200}$, however, nonlinear membership functions and standard min/max operators proved to yield most accurate results.

Table 1. Relative regression errors with 95% confidence

<table>
<thead>
<tr>
<th>Method</th>
<th>Type of Membership Function</th>
<th>Type of Aggregate Connective</th>
<th>$L_{100}$</th>
<th>$L_{150}$</th>
<th>$L_{200}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>0.0060 ± 0.0015</td>
<td>0.0044 ± 0.0014</td>
<td>0.0036 ± 0.0018</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>-</td>
<td>0.0076 ± 0.0031</td>
<td>0.0050 ± 0.0019</td>
<td>0.0044 ± 0.0022</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>0.0068 ± 0.0010</td>
<td>0.0050 ± 0.0019</td>
<td>0.0026 ± 0.0006</td>
</tr>
<tr>
<td>4</td>
<td>Linear</td>
<td>Standard</td>
<td>0.0046 ± 0.0020</td>
<td>0.0040 ± 0.0008</td>
<td>0.0040 ± 0.0042</td>
</tr>
<tr>
<td></td>
<td>Linear</td>
<td>Yager</td>
<td>0.0044 ± 0.0014</td>
<td>0.0042 ± 0.0010</td>
<td>0.0018 ± 0.0010</td>
</tr>
<tr>
<td></td>
<td>Nonlinear</td>
<td>Standard</td>
<td>0.0050 ± 0.0023</td>
<td>0.0036 ± 0.0022</td>
<td>0.0096 ± 0.0211</td>
</tr>
<tr>
<td>5</td>
<td>Linear</td>
<td>Standard</td>
<td>0.0044 ± 0.0011</td>
<td>0.0038 ± 0.0018</td>
<td>0.0022 ± 0.0005</td>
</tr>
<tr>
<td></td>
<td>Linear</td>
<td>Yager</td>
<td>0.0048 ± 0.0013</td>
<td>0.0046 ± 0.0022</td>
<td>0.0038 ± 0.0043</td>
</tr>
<tr>
<td></td>
<td>Nonlinear</td>
<td>Standard</td>
<td>0.0044 ± 0.0011</td>
<td>0.0044 ± 0.0020</td>
<td>0.0022 ± 0.0005</td>
</tr>
<tr>
<td>6</td>
<td>Linear</td>
<td>Standard</td>
<td>0.0050 ± 0.0012</td>
<td>0.0040 ± 0.0029</td>
<td>0.0026 ± 0.0014</td>
</tr>
<tr>
<td></td>
<td>Linear</td>
<td>Yager</td>
<td>0.0058 ± 0.0030</td>
<td>0.0036 ± 0.0022</td>
<td>0.0026 ± 0.0011</td>
</tr>
<tr>
<td></td>
<td>Nonlinear</td>
<td>Standard</td>
<td>0.0036 ± 0.0006</td>
<td>0.0038 ± 0.0016</td>
<td>0.0020 ± 0.0008</td>
</tr>
<tr>
<td>7</td>
<td>Linear</td>
<td>Standard</td>
<td>0.0042 ± 0.0016</td>
<td>0.0034 ± 0.0018</td>
<td>0.0098 ± 0.0209</td>
</tr>
<tr>
<td></td>
<td>Linear</td>
<td>Yager</td>
<td>0.0042 ± 0.0013</td>
<td>0.0038 ± 0.0023</td>
<td>0.0022 ± 0.0010</td>
</tr>
<tr>
<td></td>
<td>Nonlinear</td>
<td>Standard</td>
<td>0.0036 ± 0.0027</td>
<td>0.0032 ± 0.0013</td>
<td>0.0020 ± 0.0008</td>
</tr>
</tbody>
</table>
Relative regression errors are a good indicator of how an induced tree generalizes given a learning sample and a testing sample. The litmus test, nevertheless, lies in detailed examination of non-inferior solutions arrived at by a regression tree. These optimal solutions are represented by the terminal nodes of an induced tree and essentially indicate 'tight' clusters in the response region. Table 2 summarizes some of the solutions generated by each induction method on training samples of size 200 which produced the most accurate results. Furthermore, in order to verify a tree's predicted range of responses for a specific range of design variables, we employed the following technique. The objective and constraint functions were evaluated for roughly about 5000 points in an induced tree's predicted optimum design region. Means and standard deviations of the generated responses were then computed to verify the tightness of clusters which were actually formed in the predicted response region. These verified solutions appear in the last column of Table 2.

A detailed examination of Table 2 reveals that in contrast to traditional multiobjective techniques which result in distinct Pareto-optimal point-solutions, our technique identifies Pareto-optimal regions. For example, the first row of Table 2 shows four optimal solutions reported by Osyczka [15] (marked "O") which were obtained using an ordinary vector optimization technique. Clearly, each of these solutions identifies two points in the design space where objectives and constraints take their optimal values. However, in many situations either due to economical reasons or processing limitations, it is desirable to provide the designer with a range of values for design parameters where the individual variances among responses are minimal while response means are fixed on their specific optimal values.

To clarify this point, consider the first solution in Table 2 which was jointly obtained by methods 2 and 3. This solution, which was originally in the form of an if-then rule:
Table 2. Optimal solutions produced by various fuzzy and nonfuzzy splitting criteria

<table>
<thead>
<tr>
<th>Method</th>
<th>Optimum Values of Design Variables $(X_1, X_2)$</th>
<th>Predicted Pareto-Optimal $(F_1, F_2, \sigma_r)$ $10^6$ $10^3$</th>
<th>Verified Pareto-Optimal $(F_1, F_2, \sigma_r)$ $10^6$ $10^3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O$</td>
<td>$(237.0, 66.4)$</td>
<td>$(3.7 \ 0.425 \ 107.7)$</td>
<td>$(3.7 \ 0.425 \ 107.7)$</td>
</tr>
<tr>
<td></td>
<td>$(224.7, 58.6)$</td>
<td>$(4.5 \ 0.382 \ 75.3)$</td>
<td>$(4.5 \ 0.382 \ 75.3)$</td>
</tr>
<tr>
<td></td>
<td>$(235.2, 68.1)$</td>
<td>$(3.5 \ 0.437 \ 118.4)$</td>
<td>$(3.5 \ 0.437 \ 118.4)$</td>
</tr>
<tr>
<td></td>
<td>$(235.2, 70.2)$</td>
<td>$(3.3 \ 0.456 \ 137.9)$</td>
<td>$(3.3 \ 0.456 \ 137.9)$</td>
</tr>
<tr>
<td>$1$</td>
<td>$[149.6, 176.0], [68.3, 71.0]$</td>
<td>$(3.5 \ 0.438 \ 90.1)$</td>
<td>$(3.5 \pm 0.09 \ 0.438 \pm 0.007 \ 92.2 \pm 7.2)$</td>
</tr>
<tr>
<td></td>
<td>$[118.3, 149.6], [68.3, 71.0]$</td>
<td>$(3.6 \ 0.436 \ 78.8)$</td>
<td>$(3.6 \pm 0.09 \ 0.435 \pm 0.007 \ 75.8 \pm 7.0)$</td>
</tr>
<tr>
<td></td>
<td>$[152.9, 249.9], [65.6, 68.3]$</td>
<td>$(3.7 \ 0.424 \ 97.5)$</td>
<td>$(3.7 \pm 0.11 \ 0.423 \pm 0.007 \ 94.9 \pm 13.8)$</td>
</tr>
<tr>
<td>$2$</td>
<td>$[44.7, 64.8], [65.3, 75.2]$</td>
<td>$(3.7 \ 0.442 \ 39.1)$</td>
<td>$(3.7 \pm 0.30 \ 0.441 \pm 0.023 \ 35.6 \pm 10.5)$</td>
</tr>
<tr>
<td>$3$</td>
<td>$[64.8, 82.4], [69.6, 75.2]$</td>
<td>$(3.5 \ 0.456 \ 54.4)$</td>
<td>$(3.5 \pm 0.18 \ 0.458 \pm 0.016 \ 56.3 \pm 11.8)$</td>
</tr>
<tr>
<td></td>
<td>$[141.9, 219.9], [65.6, 68.3]$</td>
<td>$(3.7 \ 0.422 \ 91.7)$</td>
<td>$(3.8 \pm 0.10 \ 0.420 \pm 0.006 \ 85.2 \pm 11.2)$</td>
</tr>
<tr>
<td>$4$</td>
<td>$[10.0, 54.7], [71.0, 73.7]$</td>
<td>$(3.6 \ 0.455 \ 18.6)$</td>
<td>$(3.6 \pm 0.09 \ 0.455 \pm 0.007 \ 23.6 \pm 9.7)$</td>
</tr>
<tr>
<td></td>
<td>$[118.3, 211.7], [68.3, 71.0]$</td>
<td>$(3.5 \ 0.439 \ 93.9)$</td>
<td>$(3.5 \pm 0.11 \ 0.439 \pm 0.008 \ 93.4 \pm 16.4)$</td>
</tr>
<tr>
<td>$5$</td>
<td>$[124.8, 152.4], [68.3, 71.0]$</td>
<td>$(3.6 \ 0.436 \ 78.0)$</td>
<td>$(3.6 \pm 0.09 \ 0.436 \pm 0.007 \ 78.5 \pm 6.7)$</td>
</tr>
<tr>
<td>$6$</td>
<td>$[47.4, 102.4], [71.0, 73.7]$</td>
<td>$(3.4 \ 0.456 \ 60.7)$</td>
<td>$(3.5 \pm 0.09 \ 0.456 \pm 0.007 \ 54.8 \pm 12.5)$</td>
</tr>
<tr>
<td></td>
<td>$[124.8, 152.4], [71.0, 73.7]$</td>
<td>$(3.3 \ 0.460 \ 97.9)$</td>
<td>$(3.3 \pm 0.08 \ 0.460 \pm 0.008 \ 101.4 \pm 10.4)$</td>
</tr>
<tr>
<td>$7$</td>
<td>$[132.4, 159.9], [68.3, 71.0]$</td>
<td>$(3.6 \ 0.436 \ 82.7)$</td>
<td>$(3.6 \pm 0.09 \ 0.437 \pm 0.007 \ 82.7 \pm 6.9)$</td>
</tr>
<tr>
<td></td>
<td>$[72.4, 132.4], [68.3, 71.0]$</td>
<td>$(3.7 \ 0.434 \ 66.8)$</td>
<td>$(3.7 \pm 0.10 \ 0.434 \pm 0.006 \ 58.0 \pm 10.4)$</td>
</tr>
</tbody>
</table>
IF \((44.7 \leq X_1 \leq 64.8) \text{ AND } (65.3 \leq X_2 \leq 75.2)\) THEN

Predicted Means\((F_1, F_2, \sigma_g) = (3.7 \times 10^6, 0.442 \times 10^{-3}, 39.1)\)

recommends the design region \((X_1, X_2) = ([44.7, 64.8], [65.3, 75.2])\) where the designer can safely choose any values for design parameters \(X_1\) and \(X_2\) within the proposed bounds. The corresponding objective and constraint functions, as predicted by the learning algorithm, take the values \((3.7 \times 10^6, 0.442 \times 10^{-3}, 39.1)\) while thorough examination of roughly 5000 points in this particular design region verifies that actual responses center around the values \((3.7 \times 10^6 \pm 0.30 \times 10^6, 0.441 \times 10^{-3} \pm 0.023 \times 10^{-3}, 35.6 \pm 10.5)\) which are still well within optimal bounds (see Table 3).

<table>
<thead>
<tr>
<th>Response</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>(F_1)</td>
<td>(2.57 \times 10^6)</td>
<td>(6.56 \times 10^6)</td>
</tr>
<tr>
<td>(F_2)</td>
<td>(0.338 \times 10^{-3})</td>
<td>(0.568 \times 10^{-3})</td>
</tr>
<tr>
<td>(\sigma_g)</td>
<td>2.54</td>
<td>320.64</td>
</tr>
</tbody>
</table>

Osyczka's solutions, on the other hand, merely indicate that \((237.0, 66.4)\) or \((224.7, 58.6)\) are optimal design parameter values, and they are rigid in the sense that they fail to provide the designer with a range within which different design scenarios can be examined and subsequently realized without gross departures from optimal response regions.

In closing this section, it must be mentioned that our solutions proved adequate in that not only did they pinpoint design regions where objective functions are optimized, but they also, compared to Osyczka's solutions, maintained the bending stress \(\sigma_g\) well below its im-
posed upper bound of 180 N/mm. Furthermore, nonfuzzy partitioning techniques showed a lack of flexibility that is naturally inherent in using fuzzy techniques. The case in point is that fuzzy methods allow various approaches to the problem at hand which can affect both the quality and quantity of the solutions produced by the learning system. For example, incorporation of nonlinearities via exponential membership functions or using alternative definitions for standard min/max aggregation connectives proved beneficial in explicating more knowledge about the problem at hand.

Conclusions

In this paper we introduced a new methodology within which the problem of multiobjective optimization is transformed into induction of multivariate regression trees. Moreover, we demonstrated how the tree growing process can be accomplished by utilizing a number of concepts from diverse fields of statistics and fuzzy logic. In particular, seven splitting criteria were devised and implemented which include: three statistical methods based on dispersion matrices, two newly formulated fuzzy approaches based on Pearson’s parametric and Kendall’s nonparametric measures of association, Bellman-Zadeh’s fuzzy approach to optimization in an inductive framework, and finally, the multidimensional extension of a fuzzy measure of entropy.

We also compared the overall performance of the learning system for the fuzzy and nonfuzzy methods. Our empirical results indicate that utilization of fuzzy splitting criteria offers a degree of flexibility in terms of the learning system’s efficiency which traditional multivariate statistical methods lack. To illustrate this point, we presented the problem of multiobjective design of a beam structure.
References


FUZZY MULTIOBJECTIVE OPTIMIZATION WITH MULTIVARIATE REGRESSION TREES

A paper published in the Proceedings of the Third IEEE International Conference on Fuzzy Systems

B. Forouraghi, L.W. Schmerr, G.M. Prabhu

Abstract

In this paper we introduce a new methodology in which multiobjective optimization is formulated as unsupervised learning through induction of multivariate regression trees. In particular, it is shown that learning of Pareto-optimal solutions can be efficiently accomplished by using a number of fuzzy tree-partitioning criteria. These include: a newly formulated fuzzy method based on Kendall’s nonparametric measure of association, Bellman-Zadeh’s approach to multiobjective decision-making utilized in an inductive framework, and multidimensional fuzzy entropy. For purposes of comparison, the efficiency of learning with fuzzy partitioning criteria is compared with that of two conventional multivariate statistical techniques based on dispersion matrices. The widely-used problem of design of a three-bar truss is presented to highlight advantages of our new approach.

Introduction

Most engineering design problems require optimization of frequently conflicting objectives in the presence of multiple constraints. Consequently, the literature of multiobjective optimization has been enriched in the past three decades or so by a multitude of approaches
The main emphasis behind each of these approaches is placed upon identification of Pareto-optimal solutions by relying on precise mathematical characterizations. In many situations, however, due to the inherent fuzziness in the decision-making process, the problem at hand defies any exact formulations. For example, in a structural optimization problem the constraint may be to have the bending stress of a beam 'considerably lower' than some maximal allowable stress value. In an attempt to deal with these types of uncertainties, Bellman and Zadeh introduced a fuzzy framework within which a complex decision is viewed as a confluence of several objectives and constraints [1]. After appropriate transformation of objectives and constraints into the fuzzy domain, the optimization problem is then typically solved by nonlinear programming [11].

Naturally, multiobjective optimization, as implemented in many available techniques today, can be interpreted as an iterative searching procedure through the space of alternative solutions. Based on this observation, the artificial intelligence community has approached this problem by placing special emphasis on development of symbolic search techniques. For example, MOA*, although limited in its applicability to real-world problems, was developed as a multiobjective generalization of the heuristic search algorithm A* [2].

In this paper, we propose a new approach to multiobjective optimization which combines concepts from diverse areas such as artificial intelligence, pattern recognition, multivariate and nonparametric statistics, and fuzzy logic. In particular, it is demonstrated that multiobjective optimization can be formulated as unsupervised learning through induction of regression trees. Traditionally, tree-structured regression based on conventional statistical methods has been employed extensively in univariate regression analysis [3,13]. In the present framework, however, fuzzy multivariate extension of this type of tree induction is developed in order to construct complex multidimensional response surfaces from a number of simpler regression subsurfaces. Detailed examination of these subsurfaces, in turn,
can reveal useful information about the underlying model such as the location of the extrema and tradeoff solutions in the response region. Moreover, the tree growing process is shown to be facilitated by utilization of fuzzy logic either in isolation or within well-established statistical frameworks. Specifically, we describe three approaches: a newly formulated fuzzy clustering method based on Kendall’s nonparametric measure of association [14], Bellman-Zadeh’s decision-maximizing approach [1] to optimization in an inductive framework, and finally, the multidimensional extension of a widely-used measure of fuzzy entropy [7]. Furthermore, in order to evaluate the learning efficiency of techniques based on fuzzy logic, we also incorporated two conventional clustering criteria for tree induction which are based on dispersion matrices [5]. Our results demonstrate that, in contrast to conventional approaches to multiobjective optimization, fuzzy regression tree induction not only produces optimal solutions but also enables systematic exploration of the solution space by explicating the knowledge that actually constitutes the optimality of the generated solutions.

The remainder of this paper is organized as follows. Section 2 describes our methodology for induction of multivariate regression trees using fuzzy and nonfuzzy splitting criteria. Section 3 presents key results of applying techniques described in this paper to a multiobjective design problem. And finally, Section 4 summarizes the paper.

**Multivariate Regression Trees**

The basic element for inducing a multivariate regression tree is a set of training examples which provides a capsule view into the objective/constraint space. These examples essentially enable the learning algorithm to incrementally construct a complex regression surface from a number of simpler regression subsurfaces. This piecewise model construction is accomplished in a top-down fashion by successive partitioning of the training population at each level of the tree in an attempt to identify compact clusters in the response region. Ex-
amination of these clusters in turn can identify location of the Pareto-optimal solution where an objective can be further improved only by degrading one or more objectives [4]. The following provides more details regarding the tree induction process.

Basically, given a learning sample \( L = (X_1, Y_1), (X_2, Y_2), \ldots, (X_N,Y_N) \), the learning algorithm produces a prediction rule \( d(X):\mathbb{R}^n \rightarrow \mathbb{R}^p \) which is a mapping from the \( n \)-dimensional predictor or attribute space \( (X_i)'s \) to the \( p \)-dimensional response (objectives and constraints) region \( (Y_i)'s \). The learning sample, therefore, contains \( N \) examples where each example associates a \( p \)-dimensional response vector with an \( n \)-dimensional predictor vector. Initially, all \( N \) examples reside at the root of an empty tree. Following a divide-and-conquer approach, the root node is split into two left and right nodes such that \( n_1 \) of the original \( N \) examples fall in the left node and the remaining \( n_2 \) cases in the right node \( (N = n_1 + n_2) \). This splitting is facilitated by selection of an attribute and a threshold for partitioning the attribute's range into two regions [10]. Among all possible attribute/threshold pairs, the pair that results in the 'best' split, where the resulting left and right nodes maximize some measure of fitness, is selected and the node is split accordingly. The process of partitioning is then recursively applied to all newly generated nodes until some stopping criterion is met. In our case, a multivariate heuristic which dictates that the number of examples in a node has to be at least as large as the number of responses was used. Furthermore, after a tree is completely grown in the prescribed manner, some type of pruning will prove beneficial should the problem of overspecialization cause detrimental effects on overall efficiency of the learning system [10,3].

After the learning phase is complete, the induced tree contains a number of paths which start from the root and end in a terminal node or leaf. Each path therefore pinpoints a regression subsurface by the virtue of examples that are contained in its leaf. A leaf's set of examples can be viewed as a cluster in the response region which is characterized by its mean
vector $\mu$ and covariance $\Sigma$. The goodness of these clusters is in turn determined by a variety of statistical and fuzzy partitioning techniques which are explained below. In ensuing discussions assume that the response matrix at a given node is $R_m \times p$ which contains $m$ $p$-dimensional response vectors and that covariance of $R$ is $\Sigma_p \times p$. For fuzzy splitting criteria further assume that $R$ is converted to the multidimensional fuzzy set $M_m \times p$ by fuzzifying individual responses $r_{ij}$ in $R$ into $\mu_{ij}$ in $M$ using one of the following (user option):

1. $\mu_{ij} = \frac{(r_{ij} - \min_i \{r_j\})}{(\max_i \{r_j\} - \min_i \{r_j\})}$, $i=1,\ldots, m$ and $j=1,\ldots, p$

2. $\mu_{ij} = a \left[1 - \exp\{-b \mu_{ij}\} \right]$, $a$ and $b$ are user-defined [12]

It must be mentioned that for a given response $j$, $[\min_i \{r_j\}]$ and $[\max_i \{r_j\}]$ are found by scanning rows ($i=1,\ldots, m$) of $R$, and they can be interchanged depending upon whether the goal is to maximize or minimize the given response $j$ in the fuzzy domain.

The first two nonfuzzy splitting criteria (Methods 1 & 2) used for tree induction are the $\text{trace}(\Sigma)$ and $\text{determinant}(\Sigma)$ which denote the sum of individual response variances and the generalized variance, respectively [5]. Minimization of trace, which totally ignores the interaction among responses, attempts to locate spherically-shaped response clusters where individual variances are minimal. On the other hand, minimization of the generalized variance, $|\Sigma|$, helps identify paralleloptopes formed by response vectors which have minimal volume [15].

The next partitioning criterion (Method 3) is based on Kendall's nonparametric measure of association, $\tau$ [14]. Given the data $(R_1, S_1), \ldots, (R_N, S_N)$, $\tau_{RS}$ is given by:
which measures the amount of monotonic trend that exists between responses R and S. For example, if R and S tend to grow in a similar direction, $\tau_{RS}$ approaches 1. Conversely, if $\tau_{RS}$ approaches -1, it is concluded that R and S grow in opposite directions. Furthermore, $\tau_{RS}$ values near 0 imply absence of any monotonic relation between the two responses. Considering this, we can now incorporate elements from fuzzy logic as follows. Assume that a particular node's set M contains fuzzified responses as explained previously. Now, regardless of whether any individual response is to be maximized or minimized, the chief goal in the fuzzy domain is to locate regions where fuzzy responses approach their maximum values. Hence, given M, we obtain the matrix of correlation coefficients $T_{p \times p}$ where each $\tau_{ij}$ for responses $i$ and $j$ ($i, j = 1, \ldots, p$) is computed using Kendall's measure ($\tau_{ii} = 1, \tau_{i > j} = \tau_{i < j}$).

Note that since $T$ is symmetric, only its above-diagonal elements, $\tau_{i < j}$, are considered for further calculations. These $p(p-1)/2$ elements, which are pairwise measures of association between fuzzy responses in M, take values between -1 and 1. However, the desired clusters to be found are those for which as many of these correlation values approach 1 as possible which simply means that all or most of the responses are approaching their expected extrema in a given region. To accomplish this, T's above-diagonal $\tau_{ij}$ correlation coefficients are fuzzified using either linear or exponential membership function transformations. The aspiration levels of -1 and 1 are used in the fuzzification process to indicate that correlation values of 1 are desirable to attain maximum degree of belongingness. The cluster under consideration is then assigned the degree of trend fitness:
The overall objective, therefore, is to identify splits for which the produced clusters have maximal TF measures.

The fourth splitting criterion (Method 4) is based on Bellman-Zadeh's approach to multiobjective optimization [1]. To give a brief overview, consider making a decision $D$ which can be seen as a confluence of $n$ objectives and constraints denoted by responses $R_1, ..., R_n$. The optimal decision in the fuzzy domain then can simply be viewed as the intersection of fuzzy sets $\mu(R_1), ..., \mu(R_n)$ where each $\mu(R_i)$ is calculated using appropriate membership function transformations. More formally, the optimization task can be formulated as finding an optimum predictor vector $X^*$ for which the measure:

$$\mu_D(X^*) = \min_i \{ \mu(R_i(X)) \}$$

is maximized. Typically, after proper transformation of the problem at hand into the fuzzy domain, $X^*$ is found using nonlinear programming [11]. In our framework, however, Bellman-Zadeh's approach is used for partitioning a node such that the measure:

$$BZ(M) = [\max_i \min_j \{ \mu_{ij} \}]$$

is maximized for a particular multidimensional fuzzy set $M$ under consideration.

The last inductive partitioning technique (Method 5) to be discussed relies on fuzzy entropy [7]. Basically, given a fuzzy set $A$, fuzzy entropy $FE(A)$ measures how fuzzy actually $A$ is, where $C^O$ and $C^U$ denote counts of overlap and underlap between $A$ and $A^c$: 

$$(5) \quad TF(M) = \{\min_k \{ \mu(\tau_k) \}\}, \quad k=1, ..., p(p-1)/2$$
In a top-down inductive approach, the fuzzy entropy measure can be used to identify fuzzy clusters \( M \) which exhibit minimal amount of fuzziness at each partitioning level. The basic definition of entropy, however, has to be extended so that fuzziness of the fuzzy set \( M \) can be calculated. To accomplish this, first, \( M \)'s complement, \( M^c \) is calculated where each \( \mu_{ij}^c \) in \( M^c \) is complement of \( \mu_{ij} \) in \( M \). Then, fuzzy sets \( I_m \times p \) and \( U_m \times p \), which denote the intersection and union of \( M \) and \( M^c \), are calculated where elements \( i_{ij} \) in \( I \) and \( u_{ij} \) in \( U \) are \( \min(\mu_{ij}, \mu_{ij}^c) \) and \( \max(\mu_{ij}, \mu_{ij}^c) \), respectively. Consequently, we define the fuzzy entropy measure of a multidimensional fuzzy set \( M \) as:

\[
(9) \quad FE(M) = \left[ \max_{i=1}^{m} \min_{j=1}^{p} \{i_{ij}\} \right] / \left[ \max_{i=1}^{m} \min_{j=1}^{p} \{u_{ij}\} \right]
\]

During the course of tree induction, then, the attribute/threshold pair for which the resulting clusters have minimal fuzzy entropy are selected and the node is split accordingly.

One final comment concerning fuzzy logic-based methods 3 through 5 needs to be made at this point. The standard \( \text{min} \) and \( \text{max} \) operators are strict or pessimistic in the sense that their aggregated outputs can never exceed the highest input or be lower than the lowest input, respectively. Fuzzy set theory, however, provides a host of more optimistic aggregation connectives for integrating membership functions. In order to investigate the effects of some of these connectives on overall learning efficiency of the system, it was decided to incorporate the option of using either standard \( \text{min}/\text{max} \) operators or, alternatively, Yager’s connectives [8]:

\[
(10) \quad \max_{\text{Yager}}(\mu_1, \mu_2, ..., \mu_n) = \min(1, (\mu_1^w + \mu_2^w + ... + \mu_n^w)^{1/w})
\]
where parameter \( w > 0 \) can be varied for achieving various degrees of optimism. An interesting fact to be mentioned is that for \( w \gg 0 \), Yager's min/max operators converge to their standard definitions.

**Three-bar Truss Problem**

In this section we present the problem of optimal design of a three-bar truss which is used extensively in the literature of structural multiobjective optimization [9,11]. The design of the three-bar truss shown in Fig. 1 involves minimization of the two objectives of weight and vertical deflection of the loaded joint, where the expression for weight and deflection are:

\[
\begin{align*}
(12) \quad \text{Weight } F_1 &= 2.82X_1 + X_2 \\
(13) \quad \text{Deflection } F_2 &= (P.H)/[E.(X_1 + 1.41X_2)].
\end{align*}
\]

![Three-bar Truss](image.png)

*Fig. 1. A three-bar truss (H=1.0 and P = 20.0)*
The design variables in this example are \( X_1 \) and \( X_2 \) which represent the cross-sectional areas of the inclined and vertical bars (note that \( X_1 = X_3 \)). The overall optimization task also imposes constraints on individual stresses \( \sigma_i \)'s in bar \( i \) formulated as follows (Young's modulus, \( E = 1.0 \) and \( 0.1 < X_1, X_2 < 5.0 \)):

\[
\sigma_1 = \frac{P (X_2 + 1.41 X_1)}{1.41 X_1^2 + 2 X_1 X_2} \leq 20.0
\]

\[
\sigma_2 = \frac{P}{X_1 + 1.41 X_2} \leq 20.0
\]

\[
\sigma_3 = \frac{P X_2}{1.41 X_1^2 + 2 X_1 X_2} \leq 15.0
\]

In regards to preparation of learning and testing cases, it was decided to sample the design region in 650 distinct points between (0.1,0.1) and (5.0,5.0) in order to ensure that the response surfaces were adequately represented to the learning algorithm. Following a widely-used variance-stabilizing technique, the objective and constraint responses were transformed into log domain [6] which helped the overall learning efficiency for parametric as well as nonparametric induction criteria. The 650 points were then randomly shuffled and divided into two sets of size 450 and 200, namely, \( T_{450} \) and \( L_{200} \). The set \( T_{450} \) was dedicated entirely to testing purposes while \( L_{200} \) was used for the learning process. The learning phase then proceeded by inducing a regression tree on \( L_{200} \) for each of the fuzzy and nonfuzzy splitting methods. This entire process of random training/testing was repeated a total of five times for each tree-growing technique so that results could be represented with
95% confidence (t-distribution). It must be mentioned that for this particular example, empirical studies indicated samples of size less than 200 (i.e., 200 function evaluations) were not able to convey enough information to the learning algorithm. In other problems, however, smaller samples may provide the learning algorithm with necessary generalization power.

After completion of the learning phase, relative regression errors which normally vary between 0.0 (perfect regression model) and 1.0 (poor model) were computed for each induced tree. Our error analysis is similar to CART’s [3] except that it was extended for multivariate cases by substituting Mahalanobis distances [5] for ordinary Euclidean-based error distances in order to account for covariances that exist among responses. For nonfuzzy methods 1 and 2, the relative errors were found to be $0.028 \pm 0.01$ and $0.010 \pm 0.00$, respectively. Table 1 summarizes the results for fuzzy techniques 3 through 5.

Table 1. Relative regression errors for $L_{200}$ at 95% confidence level

<table>
<thead>
<tr>
<th>Membership function/Min-max operators</th>
<th>Method 3</th>
<th>Method 4</th>
<th>Method 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear/Standard min-max operators</td>
<td>0.012 \pm 0.005</td>
<td>0.032 \pm 0.016</td>
<td>0.026 \pm 0.018</td>
</tr>
<tr>
<td>Linear/Yager’s min-max operators</td>
<td>0.010 \pm 0.000</td>
<td>0.038 \pm 0.016</td>
<td>0.038 \pm 0.016</td>
</tr>
<tr>
<td>Nonlinear/Standard min-max operators</td>
<td>0.012 \pm 0.005</td>
<td>0.034 \pm 0.014</td>
<td>0.024 \pm 0.011</td>
</tr>
</tbody>
</table>

The performance measures shown above reveal two important facts. First, the accuracy of regression surfaces obtained through the use of fuzzy splitting criteria matches, and in few instances, surpasses the accuracy of solutions generated by well-established statistical techniques. And second, application of fuzzy splitting rules for tree induction offers a level of
flexibility which ordinary multivariate methods lack. For instance, switching from linear to nonlinear membership functions or using Yager’s min/max definitions instead of standard min/max operators frequently helped improve the quality of the obtained regression models. To illustrate this point, consider fuzzy splitting method 3. The best results for this method were attained by using a combination of linear membership functions and Yager’s min/max operators. For method 5, however, nonlinear membership functions and standard min/max operators proved to yield most accurate results.

Relative regression errors are a good indicator of how an induced tree generalizes given a learning sample and a testing sample. The litmus test, nevertheless, lies in detailed examination of non-inferior solutions arrived at by a regression tree. These optimal solutions are represented by the terminal nodes of an induced tree and essentially indicate ‘tight’ clusters in the response region. Tables 2 and 3 summarize some of the solutions generated by each induction method on training samples of size 200 which produced the most accurate results. Furthermore, in order to verify a tree’s predicted range of responses for a specific range of design variables, we employed the following technique. The objective and constraint functions were enumerated for about 5000 points in a predicted input design region. Means and standard deviations of responses were then computed to verify the tightness of clusters which were actually formed by responses in a predicted region. The predicted and verified solutions appear in the last columns of Tables 2 and 3, respectively.

A detailed examination of Tables 2 and 3 reveals that in contrast to traditional multiobjective techniques which result in distinct Pareto-optimal point-solutions, our technique identifies Pareto-optimal regions. For example, the first row of Tables 2 and 3 shows two solutions reported by Mohandas and Sandgren [9] (marked M-S) and Rao [11] (marked R) which were obtained using ordinary fuzzy and nonlinear programming techniques. Clearly, these solutions identify two points where objectives and constraints take their
Table 2. Predicted optimal solutions produced by the various techniques

<table>
<thead>
<tr>
<th>Method</th>
<th>Optimum Design Region</th>
<th>Predicted Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>($X_1, X_2$)</td>
<td>$F_1$ $F_2$ $\sigma_1$ $\sigma_2$ $\sigma_3$</td>
</tr>
<tr>
<td>M-S</td>
<td>0.59 3.27</td>
<td>4.9 3.8 18.8 3.8 15.0</td>
</tr>
<tr>
<td>R</td>
<td>0.58 3.49</td>
<td>5.1 3.6 19.1 3.6 15.4</td>
</tr>
<tr>
<td>1</td>
<td>[1.00,1.16] [1.00,3.02]</td>
<td>4.9 5.5 11.9 5.5 6.2</td>
</tr>
<tr>
<td>2</td>
<td>[0.83,1.00] [1.55,4.47]</td>
<td>5.6 3.8 12.8 3.8 8.7</td>
</tr>
<tr>
<td>3</td>
<td>[0.50,1.00] [2.40,3.09]</td>
<td>4.4 4.6 17.1 4.6 12.5</td>
</tr>
<tr>
<td></td>
<td>[0.50,1.00] [3.09,3.97]</td>
<td>5.8 3.3 14.8 3.3 11.5</td>
</tr>
<tr>
<td>4</td>
<td>[0.74,1.33] [1.66,1.99]</td>
<td>4.9 5.4 12.0 5.4 6.5</td>
</tr>
<tr>
<td></td>
<td>[0.74,1.33] [1.99,2.16]</td>
<td>5.1 4.9 11.7 4.9 6.7</td>
</tr>
<tr>
<td>5</td>
<td>[0.50,1.66] [1.83,1.99]</td>
<td>4.5 5.5 13.5 5.5 8.0</td>
</tr>
<tr>
<td></td>
<td>[0.67,1.00] [2.16,2.73]</td>
<td>4.6 4.8 14.9 4.8 10.1</td>
</tr>
</tbody>
</table>
Table 3. Verified optimal solutions produced by the various techniques

<table>
<thead>
<tr>
<th>Method</th>
<th>Optimum Design Region ($X_1, X_2$)</th>
<th>Verified Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$F_1$</td>
</tr>
<tr>
<td>M-S</td>
<td>0.59 3.27</td>
<td>4.9</td>
</tr>
<tr>
<td>R</td>
<td>0.58 3.49</td>
<td>5.1</td>
</tr>
<tr>
<td>1</td>
<td>[1.00,1.16] [1.00,3.02]</td>
<td>5.0 ± 0.5</td>
</tr>
<tr>
<td>2</td>
<td>[0.83,1.00] [1.55,4.47]</td>
<td>5.6 ± 0.8</td>
</tr>
<tr>
<td>3</td>
<td>[0.50,1.00] [2.40,3.09]</td>
<td>4.8 ± 0.4</td>
</tr>
<tr>
<td></td>
<td>[0.50,1.00] [3.09,3.97]</td>
<td>5.6 ± 0.4</td>
</tr>
<tr>
<td>4</td>
<td>[0.74,1.33] [1.66,1.99]</td>
<td>4.7 ± 0.4</td>
</tr>
<tr>
<td></td>
<td>[0.74,1.33] [1.99,2.16]</td>
<td>5.0 ± 0.4</td>
</tr>
<tr>
<td>5</td>
<td>[0.50,1.66] [1.83,1.99]</td>
<td>4.9 ± 0.9</td>
</tr>
<tr>
<td></td>
<td>[0.67,1.00] [2.16,2.73]</td>
<td>4.8 ± 0.3</td>
</tr>
</tbody>
</table>
optimal values. However, in many situations, either due to economical reasons or processing limitations, it is desirable to provide the designer with a range of values for design parameters where the individual variances among responses are minimal while response means are fixed on their specific optimal target values. To clarify this point, consider the first solution obtained by method 3. Here, the recommended design region \((X_1, X_2) = ([0.50, 1.00], [2.40, 3.09])\) enables the designer to safely choose any values for design parameters \(X_1\) and \(X_2\) within the proposed bounds. The corresponding objective and constraint functions, as predicted by the learning algorithm, take the values \((4.4, 4.6, 17.1, 4.6, 12.5)\) while thorough examination of roughly 5000 points in this particular design region verifies that actual responses center around the values \((4.8 \pm 0.4, 4.3 \pm 0.2, 15.9 \pm 2.8, 4.3 \pm 0.2, 11.5 \pm 2.6)\) which are still well within optimal bounds (see Table 4). The M-S or R solutions, on the other hand, merely indicate that \((0.59, 3.27)\) or \((0.58, 3.49)\) are optimal design parameter values, and they are rigid in the sense that they fail to provide the designer with a range within which different design scenarios can be examined and subsequently realized without gross departures from optimal response regions.

### Table 4. Range of responses for the design region \((0.1, 0.1)\) to \((5.0, 5.0)\)

<table>
<thead>
<tr>
<th>Value</th>
<th>(F_1)</th>
<th>(F_2)</th>
<th>(\sigma_1)</th>
<th>(\sigma_2)</th>
<th>(\sigma_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
<td>0.38</td>
<td>1.65</td>
<td>2.80</td>
<td>1.65</td>
<td>0.05</td>
</tr>
<tr>
<td>Maximum</td>
<td>19.14</td>
<td>82.80</td>
<td>141.40</td>
<td>82.80</td>
<td>98.60</td>
</tr>
</tbody>
</table>

In closing this section, it must be mentioned that our solutions proved adequate in that not only did they produce optimized objectives, but they also, compared to M-S and R solutions, maintained stresses \(\sigma_1\), \(\sigma_2\) and \(\sigma_3\) well below their imposed upper bounds of 20, 20 and 15, respectively. Furthermore, nonfuzzy partitioning techniques showed a lack of flexi-
bility that is naturally inherent in using fuzzy techniques. The case in point is that fuzzy methods allow various approaches to the problem at hand which can affect both the quality and quantity of the solutions produced by the learning system. For example, incorporation of nonlinearities via exponential membership functions or using alternative definitions for standard min/max aggregation connectives proved beneficial in explicating more knowledge about the problem at hand. Due to space limitations, however, Tables 2 and 3 present only some of the obtained results while the remaining solutions had to be excluded.

Conclusions

In this paper we introduced a new methodology for identifying Pareto-optimal solutions in multiobjective optimization tasks. Our approach is different from many existing techniques in that it incorporates the phenomenon of unsupervised learning into the optimization task. In particular, we showed how a multivariate regression tree can be trained to learn optimal, non-inferior solution regions using a variety of fuzzy and non-fuzzy splitting rules. These include: two statistical methods based on covariances, a newly formulated fuzzy approach based on Kendall’s nonparametric measure of association, Bellman-Zadeh’s method in an inductive framework, and finally, multidimensional fuzzy entropy in combination with minmax programming.

We also compared the overall performance of the learning system for the fuzzy and nonfuzzy methods. Our empirical results indicate that utilization of fuzzy splitting criteria offers a degree of flexibility in terms of the learning system’s efficiency which traditional multivariate statistical methods lack. To illustrate this, we presented the problem of design of a three-bar truss which is widely used in the literature of multiobjective structural optimization.
References


Abstract

In this paper we introduce a methodology within which multiobjective design optimization is approached from an entirely new perspective. Specifically, we demonstrate that multiple objective optimization through induction of multivariate regression trees is a powerful alternative to myriads of conventional vector optimization techniques which are available today. Furthermore, in an attempt to investigate the effect of various types of splitting rules on the overall performance of the optimizing system, we present a tree partitioning algorithm which utilizes a number of techniques derived from diverse fields of statistics and fuzzy logic. These include: two multivariate statistical approaches based on dispersion matrices, an information-theoretic measure of covariance complexity which is typically used for obtaining multivariate linear models, two newly-formulated fuzzy splitting rules based on Pearson's parametric and Kendall's nonparametric measures of association, Bellman and Zadeh's fuzzy decision-maximizing approach within an inductive framework, and finally, the multidimensional extension of a widely-used fuzzy entropy measure.

Finally, in order to highlight the advantages of our new methodology, we present the widely-used multiobjective design optimization of an electric discharge machining (EDM) process.
Introduction

Most engineering design problems involve optimization of several often conflicting objectives in presence of multiple constraints. The literature of multiobjective optimization has therefore been enriched since its inception in the 1960s by myriads of approaches representing diverse viewpoints and emphases from various disciplines [8]. Generally, the main goal of all these approaches is to optimize vector-valued objective functions where some or all of the objectives are most often noncommensurable. The problem of finding an optimum vector-valued objective is commonly referred to as vector optimization in the literature, and solutions of such problem are hereinafter called noninferior or Pareto-optimal as introduced by Pareto, a prominent economist at the turn of the century [9]. Determination of Pareto-optimal solutions in a given optimization task is essentially tantamount to finding a vector of optimal objectives where an individual objective can be further improved only at the cost of degrading at least one other objective [8].

Traditionally, multiple objective optimization procedures can be grouped into one of two categories: First, vector optimization techniques which rely on conventional mathematical programming (linear or nonlinear), goal programming, utility theory, etc. [18]; and second, statistical approaches such as multivariate analysis of variance (MANOVA) [17] and ellipsoidal design centering techniques [1] which are used extensively in the area of quality control. Regardless of their underlying methodological differences, however, these techniques generate Pareto-optimal solutions which generally lack two crucial characteristics. First, these solutions are represented as 'points' in both the space of design variables (independent parameters) and objective space (dependent parameters). And second, the obtained solutions are rigid in that they do not provide any understanding of the complex nature of the underlying problem which is to be solved. The following passage will clarify these points.
Relating to the first shortcoming of already-available techniques, in many real-world situations, either due to processing limitations or economic factors, it is nearly impossible to pinpoint a singular point as an optimum design vector. For example, setting a beam’s scalar design variable diameter at 3.3 mm may not be feasible either due to machine processing limitations or high degree of variability which manifests itself during the manufacturing process (e.g. 3.3 ± 0.9 mm). In fact, the main thrust behind development of the field of robust design is to determine how deviations from an optimum setting of a design vector degrade the overall performance of a system, and consequently, how to make the performance of a system least sensitive to unforeseeable deviations from recommended optimal design settings achieved through mathematical and/or statistical procedures [27].

Secondly, in terms of acquiring an understanding of the nature of the problem at hand, conventional mathematical and statistical techniques are undoubtedly capable of extracting the quantitative knowledge which specifies the static input-output behavior of a system that is to be optimized. In other words, an optimizer provides designers with a recommended set of values for input parameters of interest for which an optimal trade-off situation for objective functions is achieved. These techniques, however, do not and can not extract the knowledge which actually governs the input-output behavior of a system under examination, i.e., these techniques merely indicate what an optimum solution is but can not convey what actually constitutes the optimality of the generated solution. Hence, a particular technique has to be iterated several times under direct supervision of the user in order to obtain how deviations from one particular setting of design variables affect the overall system objectives.

Having stated the two major disadvantages of the traditional approaches to multiple objective optimization, we must now explore what other tools can potentially remedy the situation. As we mentioned, learning the optimization knowledge can have direct benefits to designers. To reiterate, the actual learning of the optimization knowledge in a given task not
only allows determination of optimal settings of design variables, but it also allows systematic examination of alternative design scenarios. The learning process, which can be defined as acquisition, assimilation and restructuring of knowledge, has received a great deal of attention from researchers in the field of artificial intelligence (AI) in the past three or four decades [11,31]. In fact, symbolic (atomistic) and subsymbolic (holistic) learning algorithms have successfully been applied to problems ranging from game playing and logistics to mass spectroscopy and design of VLSI circuitry [38]. One of the tangible benefits of applying symbolic learning algorithms, in particular, to optimization tasks is the direct acquisition of both the quantitative and qualitative knowledge which governs the behavior of a system that is to be optimized. In response to this observation, the artificial intelligence community has approached this problem by placing special emphasis on development of symbolic search techniques. For example, the MOA* algorithm, although limited in its applicability to real-world problems, was developed as a multiobjective generalization of the heuristic search algorithm A* [5].

Regardless of the range of applicability of MOA*, another artificial intelligence approach which offers a great deal of potential is the paradigm of inductive learning [23]. In an essence, inductive reasoning, first introduced by Bacon in the 1600s [11], is the highly empirical process of drawing conclusions from a given set of observed facts or data obtained through experimentation. Assuming that the input-output components of a system can be appropriately represented in the form of attribute-value design vectors, an inductive learner can discover a highly complex, and perhaps nonlinear, relationship between a set of inputs and output parameters. In fact, the process of symbolic induction (as opposed to numeric induction of subsymbolic, connectionist approaches such as neural networks) itself can be viewed as a form of optimization [29] within which concepts of interest are acquired through nu-
meric/logical processes and are then represented in symbolic forms (e.g., sentences in the first order predicate calculus) which are more in tune with human understanding.

Another paramount issue which needs to be examined in more detail is the methods of representation of optimization knowledge extracted through inductive reasoning. Learning, as we stated before, involves not only acquisition of domain-specific knowledge, but it also requires suitable methods of representing and manipulating the explicated knowledge. Some of most widely-used tools for knowledge representation are propositional calculus, first order logic and semantic networks which include decision trees, frames, and scripts [38]. Among various types of semantic networks, however, decision trees [23] and regression trees [6] have been by far the most extensively-used tools in conjunction with inductive learning. Commonly, decision trees are used in classification tasks where the correct class of an object is predicted and represented in the form of logical expressions involving a vector of inputs which describes that object in terms of its primary attributes. Regression trees, on the other hand, perform piecewise regression of continuous, complex surfaces where each leaf of an induced tree potentially identifies a simple regression subsurface. Clearly, in terms of their applicability to optimization, regression trees, in our opinion, are the most viable method of knowledge representation.

We must emphasize at this point that possibility of employing inductive learning in the area of multiobjective optimization, in general, and utilization of regression trees, in particular, have not been investigated before. The most related development to date is concerned with application of inductive, tree-structured approaches to univariate regression such as such as classification and regression trees (CART) [6] and inductive partitioning with regression trees (IPRT) [32]. These algorithms are powerful in that not only do they perform ordinary regression, but they also learn regression surfaces by extracting the knowledge that governs the input-output behavior of the model under consideration in the form of regression
trees. This form of knowledge acquisition and representation is of utmost importance if regression trees are used for performing optimization tasks. The case in point is that an induced tree essentially represents complex regression surfaces in terms of a number of simpler regression subsurfaces. Detailed examination of these subsurfaces therefore can potentially identify design regions where a product or process response is optimized. Consequently, in addition to pinpointing optimal response regions, tree-structured approaches to optimization offer the advantage of explicating the knowledge that actually constitutes the optimality of the generated solutions. This mode of behavior of regression trees is clearly advantageous over the traditional, numeric response surface methodology [34] in that it provides a direct means for capturing both the quantitative and qualitative optimization knowledge.

The main disadvantage of techniques such as CART and IPRT lies in the fact that they only address univariate regression analysis. In a typical multiobjective decision-making task, however, the optimization problem at hand consists of several design variables which in turn specify the behavior of a number of responses. For example, in a quality control application, the main objective may be to discover the underlying knowledge which controls the performance of an electric discharge machining (EDM) process in which process variables such as pulse duration and discharge current directly determine several process responses such as electrode wear, surface roughness and metal removal rate [25]. Evidently, in these situations approaches such as CART and IPRT are insufficient due to their single-response limitations.

To summarize, in this paper we present a new framework within which multiobjective optimization is accomplished through induction of multivariate regression trees. Furthermore, we present a tree partitioning algorithm which utilizes a number of inductive partitioning criteria based on concepts from statistics and fuzzy logic. Obviously, the choice of using the traditional statistical formulations in this work was instigated by the historic fact that sta-
tistics is a firmly established science with many facets which render it a particularly viable tool in many scientific applications. The theory of fuzzy sets [15,19,20], on the other hand, is a more recently developed concept, and it too has proven to be an invaluable tool in a wide array of applications ranging from pattern recognition and clustering to design of digital circuits and relational data bases [26]. In fact, within the context of multiobjective optimization, Bellman and Zadeh's fuzzy approach to optimization [3] has been widely implemented in many engineering structural optimization applications [28]. Therefore, in an attempt to examine the effects of various types of regression-tree partitioning criteria on the overall learning process which was previously explored only for decision tree algorithms [7], and also, to assess the feasibility of techniques based on fuzzy logic we describe seven splitting rules. Specifically, these include: two statistical decision rules based on dispersion matrices [35], a statistical measure of covariance complexity which is typically used for obtaining multivariate linear models [4], two newly-formulated fuzzy partitioning methods based on Pearson's parametric [17] and Kendall's nonparametric [33] measures of association, Bellman-Zadeh's decision-maximizing fuzzy approach [3] to optimization in an inductive framework, and finally, the multidimensional extension of a measure of fuzzy entropy [21].

The remainder of this paper is organized as follows. In order to make the paper self-contained, Section 2 briefly reviews the theory of fuzzy sets. Section 3 describes our methodology for transforming the problem of multiobjective optimization into induction of multivariate regression trees using fuzzy and nonfuzzy splitting criteria. Section 4 presents key results of applying techniques described in this paper to a multiobjective design problem. And finally, Section 5 summarizes the paper.
Fuzzy Logic

Classical, bivalent logic as developed originally in the antiquity by Zeno and Aristotle and extended throughout and after the scientific revolution era by G. Boole, G. Frege and B. Russell is a discipline primarily concerned with mechanization of thought processes through manipulation of propositions or symbols which are either true or false [11]. Many physical phenomena, however, defy such strict logical frameworks. For example, the set of tall men has no clear boundaries which can be expressed using bivalent logic. In fact, most observable events around us manifest a degree of uncertainty which simply cannot be represented by bivalent logic. Clearly, the set of tall men consists not only of tall and short men, but it also contains members whose degree of tallness varies gradually between the two extremes of tallness and shortness.

In order to circumvent some of the restrictions of bivalent logic, a new type of logic called fuzzy logic emerged in 1965 by L.A. Zadeh which is essentially an extension of multivalent logic as proposed in 1930 by Luckasiewicz and other mathematicians belonging to the Port Royale school of logic [20]. The main characteristic of fuzzy logic, or alternatively, fuzzy set theory, is that verity and falsehood of logical propositions vary smoothly in the interval [0,1] as opposed to bivalent logic which requires truth values be drawn from the set {0,1}. To illustrate, consider Fig. 1 which depicts three fuzzy sets A, B, and C where A represents the set of short men, B the set of men with average height of 5.9 feet, and finally C which represents the set of tall men.

As depicted in Fig. 1, the theory of fuzzy sets deals with a subset A (or B or C) of the universe of discourse X, where the transition between full membership and no membership is gradual rather than abrupt. Traditionally, the grade of membership 1 is assigned to objects that fully belong to A, while 0 is assigned to objects that do not belong to A at all.
In other words, the more an object $x$ belongs to $A$, the closer to 1 its grade of membership $\mu_A(x)$ [19]. For example, a fuzzy set such as $B$, in contrast to ordinary crisp sets, contains members whose degrees of membership or belongingness varies continuously in $[0,1]$. In particular, an individual with height of exactly 5.9 feet exhibits maximal degree of belongingness which is 1. On the other hand, as the height of an individual in set $B$ decreases from 5.9 feet, the membership value of that member in set $B$ moves farther away from 1. At the same instance, the same member's level of membership increases in fuzzy set $A$. Consequently, these overlapping sets can systematically represent various degrees of uncertainty which are normally encountered in real-world applications.

Mention must be made that assignment of membership values to individuals in a fuzzy set is a subjective, yet not an arbitrary process [36]. Essentially, two crucial properties of a characteristic or membership function $\mu_A(x)$ which assigns membership values to members $x$ in a fuzzy set $A$ govern the overall behavior of a system that is to be modeled: Its method
of acquisition and its shape. In terms of methods of acquisition, a function $\mu_A(x)$ is most often determined subjectively based on epistemological considerations (see Fig. 1) relating to the problem at hand [26]. In other instances, albeit, a characteristic function can be defined empirically based on statistical behavior of an observed system [10]. The shape of a characteristic function also plays an important role in that it allows various representations and preferences that directly relate to the underlying model. For example, sets $A$, $B$, and $C$ as depicted in Fig. 1 can respectively represent notions such as 'smaller-is-better', 'nominal-is-best', or 'larger-is-better' in an optimization task where the goal is identify members that are short, of average height, or quite tall. Furthermore, the shape of a membership function is defined by another criterion which plays an important role when dealing with incorporation of uncertainty in formal models. For instance, the set $B$ in Fig. 1 exhibits linear behavior as parameter height decreases or increases. Recent investigations however have revealed that many physical phenomena contain highly nonlinear components which can not be successfully represented by linear approximations [18,36]. Therefore, in many situations such as optimization of nonlinear processes a characteristic function may be defined which possesses nonlinear components as illustrated in Fig. 2.

![Fig. 2. Linear vs. nonlinear membership functions](image-url)
Although fuzzy logic has developed to the extent that its complete treatment is beyond scope of this work, we have now briefly presented the methods of representation and acquisition of fuzzy sets in order to make the presentation of the material in the next section self-contained (for complete details see [19]). Before proceeding further however we need to also briefly examine the fuzzy set operators which facilitate operations such as set manipulations and aggregation of evidence from multiple sources of information.

Let $A$, $B$ and $C$ be fuzzy subsets of $X$ with corresponding membership functions $\mu_A(x)$, $\mu_B(x)$ and $\mu_C(x)$, respectively. The basic operations on fuzzy sets are then defined as [15]:

A. Two fuzzy sets $A$ and $B$ are equal if and only if $\mu_A(x) = \mu_B(x)$, for all $x$ in $X$

B. $A$ is contained in $B$ if and only if $\mu_A(x) \leq \mu_B(x)$, for all $x$ in $X$

C. The union of $A$ and $B$ is a fuzzy set $C$ whose membership function is (see Fig. 3):

\[
\mu_C(x) = \max(\mu_A(x), \mu_B(x)), \text{ for all } x \text{ in } X
\]

![Fig. 3. Union of sets $A$ and $B$](image)
D. The intersection of $A$ and $B$ is a fuzzy set $C$ such that (see Fig. 4):

\[(2) \quad \mu_C(x) = \mu_A(x) \text{ AND } \mu_B(x) = \min(\mu_A(x), \mu_B(x)), \text{ for all } x \text{ in } X\]

E. The complement of $A$ is the fuzzy set $A'$ such that (see Fig. 5):

\[(3) \quad \mu_{A'}(x) = 1 - \mu_A(x), \text{ for all } x \text{ in } X\]
In closing, it must be emphasized that the intersection and union operators are by far the most important aggregate connectives, particularly in the area of decision analysis [22]. The union connective is generally used when the aggregated value is required to be high whenever any one of the input values representing different features or criteria is high. The intersection connective, on the other hand, is used when we require that the aggregated value to be high when all the inputs are high. The most popular operators which satisfy above properties for the union and intersection connectives are the max and min operators, respectively. Fuzzy set theory, however, provides a host of other connectives with slightly different properties which can be substituted for the traditional min and max operators. This point is further clarified in the following section.

**Multivariate Regression Trees**

The basic element for inducing a multivariate regression tree is a set of training examples which provides a capsule view into the objective/constraint space. These examples essentially enable the learning algorithm to incrementally construct a complex regression surface from a number of simpler regression subsurfaces. This piecewise model construction is accomplished in a top-down fashion by successive partitioning of the training population at each level of the tree in an attempt to identify compact clusters in the response region. Examination of these clusters in turn can identify location of the Pareto-optimal solution where an objective can be further improved only by degrading one or more objectives [8]. The following provides more details regarding the tree induction process.

Basically, given a learning sample $L = (X_1, Y_1), (X_2, Y_2), ..., (X_N, Y_N)$, the learning algorithm produces a prediction rule $d: R^n \rightarrow R^p$ which is a mapping from the $n$-dimensional predictor or attribute space ($X_i$'s) to the $p$-dimensional objectives and constraints space ($Y_i$'s). Fig. 6 illustrates a histogram representation of a predictor $d$. 
The learning sample, therefore, contains \( N \) examples where each example associates a \( p \)-dimensional response vector with an \( n \)-dimensional predictor vector. Initially, all \( N \) examples, denoted by population \( N(\mu, \Sigma) \) with mean vector \( \mu \) and covariance \( \Sigma \), reside at the root of an empty tree (see Fig. 7). Following a divide-and-conquer approach, the root node is inductively partitioned into two left and right nodes such that \( n_1 \) of the original \( N \) examples fall in the left node and the remaining \( n_2 \) cases in the right node (\( N = n_1 + n_2 \)).

The splitting of a parent node into two offspring nodes is facilitated by selection of an attribute and a threshold for partitioning the attribute’s range into two regions [13]. Among all possible attribute/threshold pairs, the pair that results in the 'best' split, where the resulting left and right nodes maximize some measure of fitness, is selected and the node is split accordingly.
The process of partitioning is then recursively applied to all newly generated nodes until some stopping criterion is met. In our case, in order to ensure nonsingularity of $\Sigma$, a multivariate heuristic which dictates that the number of examples in a node has to be at least as large as the number of responses was used [14]. Furthermore, after a tree is completely grown in the prescribed manner, some type of pruning will prove beneficial should the problem of overspecialization cause detrimental effects on overall efficiency of the learning system [6].

After the learning phase is complete, the induced tree contains a number of paths which start from the root and end in a terminal node or leaf. Each path therefore pinpoints a regression subsurface by the virtue of examples that are contained in its leaf. A leaf's set of examples can be viewed as a cluster in the response region which is characterized by its mean vector $\mu$ and covariance $\Sigma$. The goodness of these clusters is in turn determined by a variety of statistical and fuzzy partitioning techniques which are explained below. In ensuing discus-
sions assume that the response matrix at a given node is \( R \) (\( m \times p \) matrix) which contains \( m \) \( p \)-dimensional response vectors and that covariance of \( R \) is \( \Sigma \). For fuzzy splitting criteria further assume that \( R \) is converted to the multidimensional fuzzy set \( M \) (\( m \times p \) matrix) by fuzzifying individual responses \( r_{ij} \) in \( R \) into \( \mu_{ij} \) in \( M \) using one of the following (user option):

\[
(4) \quad \mu_{ij}^{linear} = \frac{(r_{ij} - r_{ij}^{\text{min}})}{(r_{ij}^{\text{max}} - r_{ij}^{\text{min}})} \quad i=1,\ldots,m \text{ and } j=1,\ldots,p
\]

\[
(5) \quad \mu_{ij}^{nonlinear} = a \left[ 1 - \exp\left(-b \mu_{ij}^{linear}\right) \right] \quad a \text{ and } b \text{ are user-defined [30]}
\]

It must be mentioned that for a given response \( j \), \( r_{j}^{\text{min}} \) and \( r_{j}^{\text{max}} \) are found by scanning rows \( (i=1,\ldots,m) \) of \( R \), and they can be interchanged depending upon whether the goal is to maximize or minimize the given response \( j \) in the fuzzy domain.

The first two nonfuzzy splitting criteria (Methods 1 & 2) used for tree induction are the trace and determinant of the covariance matrix which denote the sum of individual response variances and the generalized variance, respectively [12]. Minimization of trace, which totally ignores the interaction among responses, attempts to locate spherically-shaped response clusters where individual variances are minimal. Minimization of the generalized variance (\( |\Sigma| \) hereinafter), on the other hand, helps identify paralleloptopes formed by response vectors which have minimal volume [35]. To clarify this second criterion, the geometric interpretations of \( |\Sigma| \) are briefly discussed below.

Assuming that the number of observations in a multivariate population is \( p \) (number of responses), each observation vector can be represented in a space where coordinate axes correspond to individual vectors (test space). Thus, each point of test vector has two important properties. First, when scores of a vector \( X = (x_1,\ldots,x_p) \) are deviations from the mean \( X^{\text{mean}} \), the length of a test vector is:
where \( \sigma^2 \) is the variance of the test vector. And second, the cosine of the angle between any two test vectors \( X_1 \) and \( X_2 \) (\( \theta \)) is the product-moment correlation between these tests. In other words, the determinant of a simple dispersion matrix \( S \) for two responses, for example, can be written as \( \sigma_1^2 \sigma_2^2 (1-r^2) \) where \( r \) is the correlation coefficient between the two aforementioned responses. Hence, using algebraic manipulations we derive:

\[
(7) \quad \sigma_1^2 \sigma_2^2 (1-r^2) = \sigma_1^2 \sigma_2^2 (1-\cos^2 \theta) = \sigma_1^2 \sigma_2^2 \sin^2 \theta
\]

But, as shown in (6), each standard deviation is \( 1/(p-1) \) times the length of the corresponding test vector. Therefore, as illustrated in Fig. 8, for \( p = 2 \), \( |\Sigma| \) corresponds to the square of the area of the parallelogram formed by the rescaled test vectors \( X_1 \) and \( X_2 \). In the case \( p > 2 \), the generalized variance is the square of the \( p \)-dimensional volume of the parallelotope formed by the test vectors.

Fig. 8. A geometric interpretation of \( |\Sigma| \)
The third partitioning rule (Method 3) uses Bozdogan’s information-theoretic covariance complexity measure which is typically used for selection and evaluation of multivariate models [4]. Essentially, the covariance complexity metric measures how the individual subcomponents of a model or a system interact with one another. In the case of multivariate regression trees, we use a tree as a representative of an underlying model that is to be captured through the induction process. The main underlying assumption here is that a multivariate density function \( f(X) \) with a \( p \)-dimensional mean vector \( \mu \) and a \( p \times p \) positive semi-definite covariance matrix \( \Sigma \) under investigation is defined by:

\[
(8) \quad f(X) = f(x_1,\ldots,x_p) = (2\pi)^{-0.5p} |\Sigma|^{-0.5} \exp\left\{-0.5 (X-\mu)' \Sigma^{-1} (X-\mu)\right\}
\]

where the marginal entropy of \( f(X_j) \) is denoted by:

\[
(9) \quad H(X_j) = -E[\log f(X_j)] = - \int_{-\infty}^{\infty} f(x_j) \log f(x_j) dx_j
\]

and the joint entropy of \( f(X) \) is:

\[
(10) \quad H(X) = -E[\log f(X)] = - \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f(x_1,\ldots,x_p) \log f(x_1,\ldots,x_p) dx_1 \cdots dx_p
\]

As regards the information-theoretic measure of complexity, we define the quantity \( I(X) \) as the measure of interaction between random variables \( x_1,\ldots,x_p \) and express it in terms of the previously defined marginal and joint entropies as:

\[
(11) \quad I(X) = I(x_1,\ldots,x_p) = \sum_{j=1}^{p} H(x_j) - H(x_1,\ldots,x_p)
\]
Given the previous definition of joint entropy, we can then compute $H(X)$ as follows:

(12)  
$$H(X) = H(x_1, ..., x_p) = 0.5p \log(2\pi) + 0.5 \log|\Sigma| + 0.5 \sum_{j} E[(X_\mu)' \Sigma^{-1} (X_\mu)]$$

and since

(13)  
$$E[(X_\mu)' \Sigma^{-1} (X_\mu)] = E[\chi_p^2] = p$$

we easily obtain the following two expressions for the joint and marginal entropies, $H(X)$ and $H(X_j)$, respectively ($j=1, ..., p$).

(14)  
$$H(X) = H(x_1, ..., x_p) = 0.5p \log(2\pi) + 0.5 \log|\Sigma| + 0.5$$

(15)  
$$H(X_j) = 0.5 \log(2\pi) + 0.5 \log(\sigma^2_j) + 0.5$$

The total amount of interaction or informational complexity $I(X)$ is then computed as:

(16)  
$$C_o(\Sigma) = I(X_1, ..., X_p) =$$

$$\sum_{j=1}^{p} \left[ \log(2\pi) + 0.5 \log(\sigma^2_j) + 0.5 \right] - 0.5p \log(2\pi) - 0.5 \log |\Sigma| - 0.5p$$

which in turn reduces to:

(17)  
$$C_o(\Sigma) = 0.5 \sum_{j=1}^{p} \log(\sigma^2_j) - 0.5 \log |\Sigma| = 0.5 \log \prod_{j=1}^{p} (\sigma^2_j) - 0.5 \log |\Sigma|$$
But since the geometric mean of the individual variances $\sigma_1^2, \ldots, \sigma_p^2$ can be manipulated as:

\[
(18) \quad \frac{p}{2} \log \left( \prod_{j=1}^{p} (\sigma_j^2) \right)^{\frac{1}{p}}
\]

and also because of the following inequality relation between the arithmetic and geometric means of the individual variances:

\[
(19) \quad \frac{1}{p} \sum_{j=1}^{p} \sigma_j^2 \leq \left( \prod_{j=1}^{p} \sigma_j^2 \right)^{\frac{1}{p}}
\]

we can maximize the total amount of information gain $I(X)$ by rewriting $C_0(\Sigma)$ as:

\[
(20) \quad \frac{p}{2} \log \left( \frac{1}{p} \sum_{j=1}^{p} \sigma_j^2 \right) - 0.5 \log |\Sigma|
\]

The sum of variances of random variables, however, is the trace (tr) of the covariance matrix $\Sigma$. Therefore, the final expression for the measure of covariance complexity is:

\[
(21) \quad C_0(\Sigma) = \frac{p}{2} \log \left( \frac{\text{tr}(\Sigma)}{p} \right) - \frac{1}{2} \log |\Sigma|
\]
At each level of partitioning, then, a given node’s original population of responses \( R \) is divided into two subpopulations such that the covariance complexity of the resulting subpopulations are minimal. The overall task hence is to evaluate the degree of interaction that exists between responses in \( R \) and select partitions which result in minimal entropy or disorder. This can be accomplished by assigning the above measure of covariance complexity (\( C_0(\Sigma) \)) to the covariance matrix of a population \( R \) and during the tree growing process successively split a parent node into two nodes such that the measures of covariance complexity of the newly generated nodes are minimal.

The next two partitioning criteria (Methods 4 and 5) are based on Pearson’s parametric [17] and Kendall’s nonparametric [33] measures of association \( \rho \) and \( \tau \), respectively. The main motivation here is to discover the degrees of relationship between two responses \( R \) and \( S \) which may involve linear or nonlinear components. It must be emphasized that Pearson’s \( \rho \) is particularly suitable for situations where responses exhibit linear relationship. However, in many situations, linear approximations may become extremely misleading when the relationships involve nonlinear components. To this end, Kendall developed the correlation measure \( \tau \) which is not based on any parametric assumptions and is more likely to discover monotonic behavior between responses.

More formally, given the data \((R_1,S_1),\ldots,(R_N,S_N)\), Pearson’s degree of linear relationship \( \rho \) between responses \( R \) and \( S \) is:

\[
(22) \quad \rho_{RS} = \frac{\sum_{i=1}^{N} [(R_i - \bar{R})(S_i - \bar{S})]}{\sigma_R \cdot \sigma_S}
\]

where \( \sigma_R \) and \( \sigma_S \) are the standard deviations of \( R \) and \( S \), respectively. Also, Kendall’s degree of monotonic relationship between \( R \) and \( S \) is:
\[(23) \quad \tau_{RS} = \left[ \frac{2}{N(N-1)} \right] \sum_{i<j} \sum [\text{sign}(R_i - R_j) \cdot \text{sign}(S_i - S_j)]\]

where the sign function takes values +1, 0 or -1 depending upon whether its argument is positive, zero or negative. For the sake of simplicity, the following passage generically refers to \(\rho\) and \(\tau\) as \(\chi\) since the forthcoming analysis is symmetric with respect to both of these measures.

The measure of association \(\chi\) attempts to discover the relationship between any two given responses. For example, if \(R\) and \(S\) tend to grow in a similar direction, \(\chi_{RS}\) approaches 1. Conversely, if \(\chi_{RS}\) approaches -1, it is concluded that \(R\) and \(S\) grow in opposite directions. Furthermore, \(\chi_{RS}\) values near 0 imply absence of any relationship (linear in the case of \(\rho\) and monotonic in the case of \(\tau\)) between the two responses (see Fig. 9).

Considering this, we can now incorporate elements from fuzzy logic as follows. Assume that a particular node’s set \(M\) contains fuzzified responses as explained previously. Now, regardless of whether any individual response is to be maximized or minimized, the chief goal in the fuzzy domain is to locate regions where fuzzy responses approach their maximum values. Hence, given \(M\), we obtain the matrix of correlation coefficients \(T\) (p by p matrix) where each \(\chi_{ij}\) for responses \(i\) and \(j\) (\(i, j = 1, \ldots, p\)) is computed using Pearson’s or Kendall’s measures (\(\chi_{ii} = 1, \chi_{i>j} = \chi_{i<j}\)). Note that since \(T\) is symmetric, only its above-diagonal elements, \(\chi_{i<j}\), are considered for further calculations. These \(p(p-1)/2\) elements, which are pairwise measures of association between fuzzy responses in \(M\), take values between -1 and 1. However, the desired clusters to be found are those for which as many of these correlation values approach 1 as possible which simply means that all or most of the responses are approaching their expected extrema in a given region. To accomplish this, \(T\)’s above-diagonal \(\chi_{ij}\) correlation coefficients are fuzzified using either linear or exponential membership.
Fig. 9. Correlation between responses $R$ and $S$
function transformations. The aspiration levels of -1 and 1 are used in the fuzzification process to indicate that correlation values of 1 are desirable to attain maximum degree of belongingness. The cluster under consideration is then assigned the degree of trend fitness:

\[ TF(M) = \min_k \{\mu(x_k)\} \quad k=1,\ldots, p(p-1)/2 \]

The overall objective, therefore, is to identify splits for which the produced clusters have maximal TF measures.

The sixth splitting criterion (Method 6) is based on Bellman and Zadeh’s approach to multiobjective optimization [3]. To give a brief overview, consider making a decision \( D \) which can be seen as a confluence of \( n \) objectives and constraints denoted by responses \( R_1,\ldots, R_n \). The optimal decision in the fuzzy domain then can simply be viewed as the intersection of fuzzy sets \( \mu(R_1),\ldots,\mu(R_n) \) where each \( \mu(R_i) \) is calculated using appropriate membership function transformations. More formally, the optimization task can be formulated as finding an optimum predictor vector \( X^* \) for which the measure:

\[ \mu_D(X^*) = \min_i \{\mu(R_i(X))\} \]

is maximized. Typically, after proper transformation of the problem at hand into the fuzzy domain, \( X^* \) is found using nonlinear programming [28]. In our framework, however, Bellman-Zadeh’s approach is used for partitioning a node such that the measure:

\[ BZ(M) = \max_i \min_j \{\mu_{ij}\} \quad i=1,\ldots, m \text{ and } j=1,\ldots, p \]
is maximized for a particular multidimensional fuzzy set \( M \) under consideration.

The last inductive partitioning technique (Method 7) to be discussed relies on fuzzy entropy [21]. Basically, given a fuzzy set \( A \) with its complement \( A^c \), fuzzy entropy of \( A \):

\[
FE(A) = \frac{C^o(A, A^c)}{C^u(A, A^c)}
\]

measures how fuzzy actually \( A \) is, where \( C^o \) and \( C^u \) denote counts of overlap and underlap between \( A \) and \( A^c \), respectively. In a top-down inductive approach, the fuzzy entropy measure can be used to identify fuzzy clusters \( M \) which exhibit minimal amount of fuzziness at each partitioning level. The basic definition of entropy, however, has to be extended so that fuzziness of the multidimensional fuzzy set \( M \) can be calculated. To accomplish this, first, \( M \)'s complement, \( M^c \), is calculated where each \( \mu_{ij}^c \) in \( M^c \) is complement of \( \mu_{ij} \) in \( M \). Then, fuzzy sets \( I \) and \( U \) (both \( m \) by \( p \) matrices), which denote the intersection and union of \( M \) and \( M^c \), are calculated where elements \( i_{ij} \) in \( I \) and \( u_{ij} \) in \( U \) are \( \min(\mu_{ij}, \mu_{ij}^c) \) and \( \max(\mu_{ij}, \mu_{ij}^c) \), respectively. Consequently, we define the fuzzy entropy measure of a multidimensional fuzzy set \( M \) as:

\[
FE(M) = \frac{\max_i \min_j \{i_{ij}\}}{\max_i \min_j \{u_{ij}\}} \quad i=1,...,m \text{ and } j=1,...,p
\]

During the course of tree induction, then, the attribute/threshold pair for which the resulting clusters have minimal fuzzy entropy are selected and the node is split accordingly.

One final comment concerning fuzzy logic-based methods 4 through 7 needs to be made at this point. The standard \( \min \) and \( \max \) operators are strict or pessimistic in the sense that their aggregated outputs can never exceed the highest input or be lower than the lowest in-
put, respectively. Fuzzy set theory, however, provides a host of more optimistic aggregation connectives for integrating membership functions. In order to investigate the effects of some of these connectives on overall learning efficiency of the system, it was decided to incorporate the option of using either standard min/max operators or, alternatively, Yager’s operators [22]:

\[
\begin{align*}
(29) \quad & \max_{\text{Yager}}(\mu_1, \mu_2) = \min(1, (\mu_1^w + \mu_2^w)^{1/w}) \\
(30) \quad & \min_{\text{Yager}}(\mu_1, \mu_2) = 1 - \min[1, ((1 - \mu_1)^w + (1 - \mu_2)^w)^{1/w}] \\
(31) \quad & \text{complement}_{\text{Yager}}(\mu) = (1 - \mu^w)^{1/w}
\end{align*}
\]

where parameter $w > 0$ can be varied for achieving various degrees of optimism. An interesting fact to be mentioned is that for $w \gg 0$, Yager’s min/max connectives converge to their standard definitions.

A Design Example

In this section we present a canonical example relating to process optimization of an electric discharge machining (EDM) procedure [25]. Before discussing the actual results, however, three crucial issues which are directly related to performance evaluation of a learning system must be closely examined: (1) the resampling technique for preparation of training and testing cases, (2) variance-stabilization for parametric splitting rules which rely exclusively on at least the assumption of normal distribution of the underlying observed data vectors, and (3) the regression error analysis.
Resampling Techniques

An important step towards construction and evaluation of a learning system concerns with the preparation of training and testing samples. An obvious solution is to perform the training and testing procedures on the same set of data and compute the apparent error rate which can be a highly overoptimistic estimate of a learning system's performance. A statistically rigorous technique for producing better error estimates is random subsampling. In this technique, a data set is divided into multiple random train-and-test subsets and the overall error estimate is essentially the average of the error rates for regression models derived for the independently and randomly generated training-and-testing partitions [37].

In regards to preparation of training and testing cases for the EDMP example that will be discussed later, the design region \((X_1, X_2)\) was sampled in 900 distinct points between \((a,b)\) and \((c,d)\) in order to ensure that the response surfaces were adequately represented to the learning algorithm. Note that the design variables \(X_1\) and \(X_2\) vary respectively in closed intervals \([a,c]\) and \([b,d]\). The 900 points were then randomly shuffled and divided into two sets of size 450 each, namely, \(T_{450}\) and \(L_{450}\). The set \(T_{450}\) was dedicated entirely to testing purposes while \(L_{450}\) was used for the learning process. Samples of size 100, 150 and 200 were then randomly drawn from the overall learning set \(L_{450}\). The learning phase then proceeded by inducing a regression tree on each of the randomly drawn samples \(L_{100}\), \(L_{150}\) and \(L_{200}\) for each of the fuzzy and nonfuzzy splitting methods. This entire process of random selection of training samples, learning and testing was repeated a total of five times for each tree-growing technique so that results could be represented with 95% confidence (t-distribution). It must be mentioned that for this particular example, empirical studies indicated samples of size less than 100 (i.e., 100 function evaluations) were not able to convey
enough information to the learning algorithm. In other problems, however, smaller samples may provide the learning algorithm with the necessary generalization power.

**Variance-Stabilization**

When building linear models, a crucial consideration is with the effects of violations of distributional assumptions in multivariate statistical analysis [16]. All of the parametric splitting rules used in this work are derived under at least the assumption of multivariate normality which ensures that the data vectors are independent random samples from a population in which any linear combination of the variables in the data vector is normally distributed [2]. In reality however it may not be possible to satisfy the normality requirement due to the skewed distributional nature of the gathered data. For example, the arithmetic mean of a population may not necessarily correspond to the point where 68% of the population falls within one standard deviation of the mean. The common solution for this situation is to use two types of variance stabilization techniques: Log transformation and rank transformation.

The log transformation is a powerful technique which *smoothes* the underlying data distribution in such a way that the log of a non-normal response is more likely to follow or approach normal distribution while the original response will then have a lognormal distribution [16]. Furthermore, aside from elimination of some of the nonlinear components of the original responses, the log transformed data causes the arithmetic means to be converted to geometric means which are better indicators of central tendencies of a population. Mention must be made that log transformation are routinely used in the field of optimization especially in the area of quality control [27].

The rank transformation procedures, on the other hand, are ones in which the usual parametric procedure is applied to the ranks of the data instead of the data themselves [39].
The rank transformed data therefore better approximate normal distribution due to the fact that ranks are drawn from a population (natural numbers) where individual members have equal probability of occurrence. Another advantage of rank transformed data is that instead of employing nonparametric tests of significance, parametric models can still be employed while the underlying data are readily represented in terms of their respective ranks.

Mention must be made that in this work both types of power transformation, namely log and rank transformations, were attempted. The empirical results obtained indicated that log transformation in general produced far more accurate results than the rank-ordered data, and therefore, the results presented later in this section are entirely based on stabilizations achieved while performing learning in the log domain.

Regression Error Analysis

The error analysis used for the multivariate regression tree models in this work is based on CART's approach [6]. Essentially, given a learning sample $L$ consisting of $(x_1, y_1), \ldots, (x_N, y_N)$, a predictor $d(x)$ has a corresponding error estimate $R^*(d)$. Assuming that $L$ is divided into training set $L_1$ and testing set $L_2$ which respectively contain $N_1$ and $N_2$ examples, CART computes the error of the predictor $d(x)$ which is built on $L_1$ as:

$$ R^*(d) = \frac{1}{N_2} \sum (y_n - d(x_n))^2 $$ for all $x_n$ and $y_n$ in $L_2$.

The mean squared error for regression however depends on the scale in which the response is measured. For this reason, a normalized measure of accuracy $R^*(\mu)$ which removes the scale dependence is often used. The idea here is that $\mu$ is the baseline predictor for $Y$ if nothing is known about $X$. Then, the performance of any predictor $d$ based on $X$ can be
normalized by computing a relative error term $RE^*(d)$:

\[(33) \quad RE^*(d) = R^*(d)/R^*(\mu)\]

The relative error is always nonnegative and is usually, but not always, less than 1. Most sensible predictors $d(x)$ are more accurate than $\mu$ and hence result in $RE^*(d) < 1$.

In order to accommodate this type of analysis in multivariate linear models, the following extensions were implemented. As mentioned before, an overall regression surface in a regression tree approach is represented by a number of simpler regression subsurfaces which are represented by leaves of the tree along various paths. Let's assume that during the testing phase of an induced tree a test case $X$ falls in a leaf identified by a normal population $N(\mu, \Sigma)$. The weighted Euclidean distance between vector $X$ and population $N$, namely $\Delta(X,N)$, is called the *Mahalanobis* distance [12] which in an essence takes into account the covariance of the identified population $N$ at a given leaf. The distance $\Delta(X,N)$ is defined as:

\[(34) \quad \Delta^2(X,N) = (X - \mu)' \Sigma^{-1} (X - \mu)\]

Hence, we express $R^*(d)$ as:

\[(35) \quad R^*(d) = \Delta(X,N)\]

and the mean estimate as:

\[(36) \quad R^*(\mu) = (y - \mu)' \Sigma^{-1} (y - \mu) \quad \text{for all } y \text{ in population } N(\mu, \Sigma)\]

and compute the relative regression error $RE^*(d)$ in the manner prescribed earlier.
The EDM Process

The last example presented in this section concerns with optimization of an electric discharge machining (EDM) process [25]. In this process the main input quantities are pulse duration ($T_i$), pulse interval ($T_o$), amplitude of the discharge current ($I$), erosion diameter ($\phi$) and erosion depth ($g$). The optimization process for the four objective functions can be formally stated as:

Objectives:

Maximize metal removal rate, $Q_v$ \( \text{mm}^3/\text{min} \)

Minimize electrode wear, $\delta$ \( \% \)

Minimize machine power consumption, $N$ \( W \)

Minimize surface roughness, $R_a$ \( \text{mm} \)

where dependencies between input and output quantities were experimentally determined through a series of statistical regression analyses with the following results:

\[
Q_v = e^{11.744 - 1.555 + 0.047 \ln T_i + 0.276 \ln \phi + 0.051 \ln g - 0.174 \ln \phi - 0.107 + 0.155 \ln g - 0.107 - 0.124 \ln g_{-0.742}}
\]

\[
\delta = e^{-81.509 + 5.634 - 0.349 \ln T_i - 0.335 \ln T_o + 0.119 \ln \phi + 0.174 \ln g}
\]

\[
T_i = 3.726 - 0.551 \ln T_i - 0.344 \ln T_o + 0.253 \ln g - 0.207 \ln \phi + 13.609 + 0.207 \ln g - 0.207 \ln g_{-3.102}}
\]

\[
\phi = 12.219 - 0.71 \ln g_{-3.102}}
\]
In order to realistically reduce the size of the search space in this problem, it is assumed that the selection of the optimal machining conditions is fixed for the workpiece whose diameter $\phi = 68$ mm and depth $g = 6$ mm. Furthermore, the three remaining three input quantities vary as follows:

$$500 \leq T_i \leq 2000$$
$$64 \leq I \leq 128$$
$$125 \leq T_0 \leq 250$$

The preparation of the training and testing samples in this example was performed identical in the manner described for the first two examples. After completion of the learning phase, the relative regression errors with 95% confidence were computed which appear in Table 1.

The predicted and verified Pareto-optimal solutions along with the range of responses for the four objective functions in the EDM process are also respectively summarized in Ta-
bles 2, 3 and 4. Note that the first solution appearing in Tables 2 and 3 was reported by Osyczka and was obtained using an ordinary vector optimization technique. Again, our solutions are superior to conventional solutions in that the design region with its associated tolerance regions offers a degree of flexibility during the design process which other techniques clearly lack.

Table 1. Relative regression errors for EDMP example

<table>
<thead>
<tr>
<th>Method</th>
<th>$L_{100}$</th>
<th>$L_{150}$</th>
<th>$L_{200}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.046 ± 0.0068</td>
<td>0.042 ± 0.0055</td>
<td>0.038 ± 0.0103</td>
</tr>
<tr>
<td>2</td>
<td>0.058 ± 0.0055</td>
<td>0.056 ± 0.0068</td>
<td>0.050 ± 0.0087</td>
</tr>
<tr>
<td>3</td>
<td>0.007 ± 0.0022</td>
<td>0.005 ± 0.0034</td>
<td>0.004 ± 0.0026</td>
</tr>
<tr>
<td>4</td>
<td>0.044 ± 0.0068</td>
<td>0.026 ± 0.0068</td>
<td>0.022 ± 0.0103</td>
</tr>
<tr>
<td>Linear</td>
<td>0.024 ± 0.0111</td>
<td>0.020 ± 0.0087</td>
<td>0.012 ± 0.0055</td>
</tr>
<tr>
<td>MinMax</td>
<td>0.030 ± 0.0000</td>
<td>0.024 ± 0.0141</td>
<td>0.010 ± 0.0000</td>
</tr>
<tr>
<td>Yager</td>
<td>0.044 ± 0.0103</td>
<td>0.032 ± 0.0055</td>
<td>0.024 ± 0.0068</td>
</tr>
<tr>
<td>5</td>
<td>0.046 ± 0.0111</td>
<td>0.032 ± 0.0055</td>
<td>0.018 ± 0.0055</td>
</tr>
<tr>
<td>Linear</td>
<td>0.046 ± 0.0111</td>
<td>0.032 ± 0.0055</td>
<td>0.022 ± 0.0055</td>
</tr>
<tr>
<td>MinMax</td>
<td>0.024 ± 0.0207</td>
<td>0.021 ± 0.0143</td>
<td>0.009 ± 0.0005</td>
</tr>
<tr>
<td>Yager</td>
<td>0.024 ± 0.0111</td>
<td>0.016 ± 0.0068</td>
<td>0.009 ± 0.0005</td>
</tr>
<tr>
<td>6</td>
<td>0.0010 ± 0.0000</td>
<td>0.010 ± 0.0067</td>
<td>0.008 ± 0.0011</td>
</tr>
<tr>
<td>Linear</td>
<td>0.030 ± 0.0196</td>
<td>0.028 ± 0.0296</td>
<td>0.024 ± 0.0111</td>
</tr>
<tr>
<td>MinMax</td>
<td>0.022 ± 0.0103</td>
<td>0.012 ± 0.0055</td>
<td>0.009 ± 0.0111</td>
</tr>
<tr>
<td>Yager</td>
<td>0.010 ± 0.0000</td>
<td>0.010 ± 0.0067</td>
<td>0.008 ± 0.0015</td>
</tr>
</tbody>
</table>
Table 2. Predicted optimal solutions generated for EDMP example

<table>
<thead>
<tr>
<th>Method</th>
<th>Optimum values of design variables</th>
<th>Predicted Pareto-Optimal $Q_v$</th>
<th>$\delta$</th>
<th>$N$</th>
<th>$R_n \times 10^{-3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Osyczka</td>
<td>(64, 2000, 125)</td>
<td>2458</td>
<td>0.24</td>
<td>3727</td>
<td>0.376</td>
</tr>
<tr>
<td>1, 2</td>
<td>[64,72] [1732,2000] [237,250]</td>
<td>2376</td>
<td>0.22</td>
<td>3709</td>
<td>0.164</td>
</tr>
<tr>
<td>1</td>
<td>[64,69] [1732,2000] [237,250]</td>
<td>2376</td>
<td>0.22</td>
<td>3709</td>
<td>0.164</td>
</tr>
<tr>
<td>3</td>
<td>[64,72] [1732,2000] [212,250]</td>
<td>2433</td>
<td>0.23</td>
<td>3758</td>
<td>0.170</td>
</tr>
<tr>
<td></td>
<td>[64,69] [1732,2000] [162,187]</td>
<td>2475</td>
<td>0.24</td>
<td>3774</td>
<td>0.253</td>
</tr>
<tr>
<td>4</td>
<td>[64,77] [1732,2000] [237,250]</td>
<td>2559</td>
<td>0.24</td>
<td>3874</td>
<td>0.167</td>
</tr>
<tr>
<td></td>
<td>[64,73] [1732,2000] [187,212]</td>
<td>2508</td>
<td>0.24</td>
<td>3815</td>
<td>0.217</td>
</tr>
<tr>
<td>5, 6</td>
<td>[64,77] [1732,2000] [162,187]</td>
<td>2642</td>
<td>0.26</td>
<td>3930</td>
<td>0.258</td>
</tr>
<tr>
<td>5</td>
<td>[64,69] [1732,2000] [212,237]</td>
<td>2459</td>
<td>0.23</td>
<td>3777</td>
<td>0.187</td>
</tr>
<tr>
<td>6</td>
<td>[64,67] [1732,2000] [162,250]</td>
<td>2396</td>
<td>0.22</td>
<td>3715</td>
<td>0.201</td>
</tr>
<tr>
<td>7</td>
<td>[64,67] [1732,2000] [162,187]</td>
<td>2475</td>
<td>0.24</td>
<td>3774</td>
<td>0.253</td>
</tr>
</tbody>
</table>
Table 3. Verified optimal solutions generated for EDMP example

<table>
<thead>
<tr>
<th>Method</th>
<th>Optimum values of design variables $(I,T_1,T_0)$</th>
<th>Verified Pareto-Optimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>$(64,2000,125)$</td>
<td>$Q_v$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2458</td>
</tr>
<tr>
<td>1</td>
<td>[64,72] [1732,2000] [237,250]</td>
<td>$Q_v$</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>2474 ± 139</td>
</tr>
<tr>
<td>1</td>
<td>[64,69] [1732,2000] [237,250]</td>
<td>$Q_v$</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>2393 ± 91</td>
</tr>
<tr>
<td>3</td>
<td>[64,72] [1732,2000] [212,250]</td>
<td>$Q_v$</td>
</tr>
<tr>
<td>4</td>
<td>[64,69] [1732,2000] [162,187]</td>
<td>$Q_v$</td>
</tr>
<tr>
<td>5</td>
<td>[64,77] [1732,2000] [237,250]</td>
<td>$Q_v$</td>
</tr>
<tr>
<td>6</td>
<td>[64,73] [1732,2000] [187,212]</td>
<td>$Q_v$</td>
</tr>
<tr>
<td>5</td>
<td>[64,77] [1732,2000] [162,187]</td>
<td>$Q_v$</td>
</tr>
<tr>
<td>6</td>
<td>[64,69] [1732,2000] [212,237]</td>
<td>$Q_v$</td>
</tr>
<tr>
<td>6</td>
<td>[64,67] [1732,2000] [162,250]</td>
<td>$Q_v$</td>
</tr>
<tr>
<td>7</td>
<td>[64,67] [1732,2000] [162,187]</td>
<td>$Q_v$</td>
</tr>
</tbody>
</table>

Example values:
- Method 1:
  - $Q_v = 2474 \pm 139$
  - $\delta = 0.27 \pm 0.02$
  - $N = 3766 \pm 128$
  - $R_a \times 10^{-3} = 0.181 \pm 0.008$
- Method 2:
  - $Q_v = 2393 \pm 91$
  - $\delta = 0.25 \pm 0.02$
  - $N = 3693 \pm 87$
  - $R_a \times 10^{-3} = 0.179 \pm 0.008$
Table 4. Range of responses in the design region (64, 500, 125) to (128, 2000, 250)

<table>
<thead>
<tr>
<th>Response</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_v$</td>
<td>1994</td>
<td>6797</td>
</tr>
<tr>
<td>$\delta$</td>
<td>0.21</td>
<td>13.14</td>
</tr>
<tr>
<td>$N$</td>
<td>2838</td>
<td>7202</td>
</tr>
<tr>
<td>$R_a$</td>
<td>0.000161</td>
<td>0.0012</td>
</tr>
</tbody>
</table>

Conclusions

In this paper we introduced a new methodology within which the problem of multiobjective optimization is transformed into induction of multivariate regression trees. One of the main advantages of our approach to optimization, in contrast to a host of other conventional techniques, is that it not only identifies Pareto-optimal solutions, but it also explicates the knowledge which establishes the optimality of the generated solutions. This added capability is crucial during the design process where feasibility of various design scenarios often has to be explored before a final decision is made. Moreover, we demonstrated how the tree growing process through inductive partitioning can be accomplished by utilizing a number of concepts from diverse fields of statistics and fuzzy logic. In particular, seven splitting criteria were devised and implemented which include: two statistical methods based on dispersion matrices, an information-theoretic measure of covariance complexity which is typically used for obtaining multivariate linear models, two newly formulated fuzzy approaches based on Pearson's parametric and Kendall's nonparametric measures of association, Bellman-Zadeh's fuzzy approach to optimization in an inductive framework, and finally, the multidimensional extension of a fuzzy measure of entropy.
We also compared the overall performance of the learning system for the fuzzy and nonfuzzy methods. Our empirical results indicate that fuzzy inductive partitioning criteria used during the tree-growing process offer a degree of flexibility in terms of the learning system's efficiency which traditional multivariate statistical methods lack. To illustrate this point, we presented a multiobjective design problem which dealt with optimization of an electric discharge machining (EDM) process.

References


GENERAL CONCLUSIONS

In this dissertation we have presented six papers which discuss a number of issues and application areas relating to the topic of inductive learning of decision trees and regression trees.

In the first paper, a new approach to incremental construction of ID3-type decision trees was described. The main motivation behind conception of our proposed solution, called IDea, is that previous solutions to incremental learning such as ID4 and ID5 algorithms were seen inadequate due to their instance-by-instance type of approach to incremental learning. In contrast, IDea uses incremental batch learning which can essentially be viewed as learning from multiple batches containing multiple examples. One advantage of batch learning is that learning in noisy domains is shown to be more accurate if it is tuned for more than one example at a time. Furthermore, we demonstrated how IDea's storage requirements, and hence, its computational complexity can greatly be reduced by adopting instance selection techniques commonly used in instance-based learning (IBL) algorithms. The experiments we conducted show that IDea's performance is comparable to, and in some instances better than, that of single-batch ID3 algorithm while using notably fewer training instances.

In the second paper we introduced a new technique for reducing the dimensionality of a given set of training examples along the two dimensions of instance space and feature or attribute space. The rationale behind development of our technique lies in the fact that in most real-world applications, in the absence of any universal criteria for preparation of appropriate training information, inductive machine learning systems are often inundated with multitudes of examples which can contain superfluous information. The presence of these data redundancies in turn can degrade the performance of an inductive learner such as a neural network. Our proposed methodology here effectively eliminates such redundancies by creating a synergy whereby incremental batch learning of an inductive decision tree algorithm is util-
ized to select a subset of an original set of training data which is deemed as most informative. Our empirical results on the problem of classification/detection of hard-alpha inclusions in titanium alloys indicate that inductive neural learning of a standard connectionist network can generally benefit from such data reductions in two ways: First, in terms of improvements in the classification accuracy of the inductive learner; and second, in terms of savings in the amount of time that is needed to train the network.

In the third paper we made the observation that with the rapid pace of growth in engineering design and manufacturing, it is crucial for designers, who may not be expert process or analysis engineers, to have a working knowledge of materials selection and performance issues. In absence of real materials experts, however, searching for various types of materials information is a time-consuming task which requires extensive examination of product sheets, catalogs, reference books, or property databases. Consequently, our solution to this problem is to use an ID3-type decision tree learning system that can automatically discover knowledge of materials selection from a commercially available property database. This knowledge is represented in the familiar form of IF-THEN rules which are readily made available to designers. Moreover, we showed that our developed ID3-type classifier is suitable for real world-applications where training materials information may be imprecise/missing or become available incrementally over time.

In the last three papers we introduced a novel methodology within which the problem of multiobjective optimization is transformed into induction of multivariate regression trees. Moreover, we demonstrated how the tree growing process can be accomplished by utilizing a number of concepts from diverse fields of statistics and fuzzy logic. In particular, seven splitting criteria were devised and implemented which include: three statistical methods based on dispersion matrices, two newly formulated fuzzy approaches based on Pearson’s parametric and Kendall’s nonparametric measures of association, Bellman-Zadeh’s fuzzy
approach to optimization in an inductive framework, and finally, the multidimensional ex-
tension of a fuzzy measure of entropy. We also compared the overall performance of the 
learning system for the fuzzy and nonfuzzy methods. Our empirical results indicate that 
utilization of fuzzy splitting criteria offers a degree of flexibility in terms of the learning sys-
tem's efficiency which traditional multivariate statistical methods lack. To illustrate this 
point, we presented three examples which respectively deal with multiobjective design of a 
three-bar truss, a beam, and an electric discharge machining (EDM) process.
I would like to express my sincere gratitude to my academic advisors Dr. Gurpur Prabhu and Dr. Lester Schmerr for their support and guidance in this work as well as in my graduate studies. Thanks are also extended to Dr. S. Kothari, Dr. V. Honavar and Dr. E. Bartlett for serving on my committee.

I would like to thank my parents, Ali and Homa, for their constant encouragement and support throughout every stage of my life and also my family here, Kaveh, Amy, Aria, Farhad and Lailei, for their invaluable console and helpful suggestions.

Thanks are also due to my friend and colleague Chien-P. Chiou for his assistance and many valuable discussions.

This work was supported by the Center for Nondestructive Evaluation (CNDE) at Iowa State University, Ames, Iowa.