SUBMITTED TO IEEE EXPERT, March 1991.

# Machine Learning of Expert System Rules: Applications to Semiconductor Manufacturing

**Keki B. Irani** **Jie Cheng** **Usama M. Fayyad** **Zhaogang Qian**

*Artificial Intelligence Laboratory*
*Electrical Engineering and Computer Science Department*
*The University of Michigan*
*Ann Arbor, MI 48109-2122*
*(313) 764-8517*

Irani@caen.engin.umich.edu

**Abstract**

This paper introduces a machine learning technique for aiding automated knowledge acquisition. We motivate the problem of learning classification rules from data and argue its necessity in certain industrial settings. After introducing the learning algorithm, we review some successful applications in semiconductor manufacturing. Finally we introduce two extensions to our algorithm to deal with problems encountered in industry data: noise and limited training sets.

We argue the appropriateness and necessity of machine learning to circumvent the "knowledge acquisition bottleneck", and motivate and describe the learning algorithm we developed. The GID3 system was applied to five different projects with several Semiconductor Research Corp. (SRC) industry members. We describe some of the application areas where acceptable levels of success were achieved by the program. The application areas include: identification of relationships between Reactive Ion Etching (RIE) process anomalies and the corresponding parameter settings, acquisition of a set of rules for RIE process optimization, and knowledge acquisition for an emitter piloting advisory expert system. The paper aims to bring attention to machine learning as a useful tool in the automation of the semiconductoring manufacturing process and as an aid to engineers in interpreting and assimilating experimental results.

**Keywords:** Machine learning, automated knowledge acquisition, decision tree induction, process diagnosis, classification.

# 1 Introduction

There is a steadily increasing drive towards automation in all aspects of human endeavor. Automation promises cost-effectiveness, reliability, predictability, and accuracy. So far, only fixed simple tasks in manufacturing have been automated. Automation of more complex tasks, currently requiring intelligent decision making or problem solving on the part of humans, is a much more difficult task.

One of the goals of Artificial Intelligence (AI) research is to provide mechanisms for emulating human decision-making and problem solving capabilities, using computer programs. The first AI attempts at such systems appeared as part of the technology known as "expert systems". Expert systems are intended to provide the means of encoding human knowledge about a specific task in terms of situation-action rules. The idea is that if such systems are endowed with sufficient knowledge of the task at hand, they may be able to emulate human expert behavior in most, if not all, situations that arise during task execution.

Even with such a constrained and narrow goal, serious difficulties arose that hindered the development of successful expert system applications. The first such difficulty is known as the "knowledge acquisition bottleneck" [Fieg81]. Human experts find it difficult to express their knowledge, or explain their actions, in terms of concise situation-action rules. If pressed to do so, they typically produce rules that are incorrect, or that have many exceptions. The articulation of specific intuitive knowledge into deterministic rules is a difficult, sometimes unrealistic, problem for human experts. Interviewing domain experts to extract such knowledge is also an expensive process demanding time from experts and knowledge engineers. In addition, it is a difficult and often frustrating process for the domain experts involved. Industrial diagnostic expert systems typically require a long development time.

A second problem arises in a different situation: What if a task is not well-understood, even by the experts in that area? An example of this situation is manifested in our experience with the automation of the reactive ion etching (RIE) process in semiconductor manufacturing. We discuss some of the details of this application later in the paper. In such domains, abundant data are available from the experiments conducted, or items produced. However, models that relate

how output variables are affected by changes in the controlling variables are not available. Experts strongly rely on familiarity with the data and on "intuitive" knowledge of such a domain. How would one go about constructing an expert system for such a domain?

## 1.1 The Machine Learning Approach

The machine learning approach to circumventing the aforementioned hurdles calls for extracting classification rules from data directly. Rather than require that a domain expert provide domain knowledge, the learning algorithm attempts to discover, or induce, rules that emulate expert decisions in different circumstances by observing examples of expert tasks.

A training example consists of a description of a situation and the action performed by the expert in that situation. The situation is described in terms of a set of *attributes*. An attribute may be *continuous* (numerical) or *discrete* (nominal). For example, a nominal attribute may be *shape* with values { *square, triangle, circle*}. An example of a continuous attribute is pressure or temperature. The action associated with the situation, the *class* to which the example belongs, is a specification of one of a fixed set of allowed actions. The class of each training example is typically determined by a human expert during normal execution of his/her task. Example actions may be *raise temperature, decrease pressure, accept batch,...* The goal of the learning program is to derive conditions, expressed in terms of the attributes, that are predictive of the classes. Such rules may then be used by an expert system to classify future examples. Of course, the quality of the rules depends on the validity of the conditions chosen to predict each action.

A training example is therefore a list of the values of all the attributes along with the class to which the example belongs. Assume there are $m$ attributes $A_1, \ldots, A_m$, $p$ classes $C_1, \ldots, C_p$. A training example is an $m+1$-tuple $\langle b_1, b_2, \ldots, b_m; C_j \rangle$, where each $b_i$ is one of the values of the attribute $A_i$: $\{a_{i1}, \ldots, a_{ir_i}\}$, and $C_j$ is one of the $p$ classes. A rule for predicting some class $C_j$ consists of a specification of the values of one or more attributes on the left hand side and that class on the right hand side.

As an example, consider the simplified small example set shown in Table 1. This set consists of six examples e-1 through e-6. There are two attributes: *selectivity* and $\Delta$ *line*

| example | Selectivity | Δ line width | class |
|---|---|---|---|
| e-1 | normal | normal | *power is high* |
| e-2 | normal | high | *power is low* |
| e-3 | high | high | *power is low* |
| e-4 | high | low | *power is high* |
| e-5 | low | normal | *flow rate is low* |
| e-6 | low | high | *flow rate is low* |

Table 1: A Simple Training Set of Examples.

*width.* The attributes can take the values *low, normal,* and *high.* There are three classes: *flow rate is high, power is low,* and *power is high.* A simple rule consistent with these examples may be:

IF (Selectivity = low) THEN *Flow rate is low*

Note that this is only an illustrative simplification. Typically, the number of examples of a meaningful training set is at least in the hundreds, while the number of attributes is usually in the tens.

## 1.2 Further Motivation for Machine Learning

In addition to the motivations listed above, two other reasons exist for the need of a machine learning approach. The first is the growing number of large databases that store instances of diagnostic tasks. Such data is typically accessed by keyword or condition lookup. As the size of the database grows, such an approach becomes less effective. Suppose an expert needs to look up cases similar to a case being diagnosed. A query may easily return hundreds of matches. A method for determining relevant conditions automatically would be needed in this case.

Another motivation is the evolution of complex systems that have an error detection capability. Communication networks are an example. Faults are detectable by the network hardware. Several thousand faults may occur during a day. To debug such a network, a human would need to sift through large amounts of data in search of an explanation. An automated capability

of deriving conditions under which certain faults occur may be of great help to the engineer in uncovering underlying problems in the hardware.

Both of the above situations indicate that machine classification learning is a potentially powerful method for summarizing large amounts of data effectively.

### 1.3 Industrial Applications and Problems

There are several approaches to inducing diagnostic rules from data. In this paper we do not cover all the details, nor do we review the relevant machine learning literature. We restrict our discussion to briefly presenting the problem and its complexity, and then we focus our attention on the induction of decision trees as an efficient solution. We illustrate this discussion with simple examples. We then briefly motivate and outline our algorithm (GID3) for inducing decision trees. The second part of the paper provides some details of several industrial applications in semiconductor manufacturing domains for which GID3 was utilized and was found useful by the process and knowledge engineers.

We also cover two problems that we faced in our dealings with industrial data and the schemes we developed to overcome them. The typical assumption is that large amounts of data are available when machine learning is to be applied. However, there are cases when experiments may be very expensive. In such cases, training data are limited. We developed a system, KARSM, that uses Response Surface Methodology, coupled with GID3, to generate rules under such conditions. Another problem we face with industrial data is that in some processes the data may be noisy. Human recording errors, limited sensor resolution, or sensor or equipment reliability problems introduce inaccuracies in the values of the attributes. We developed a system, RIST, that utilizes statistically robust techniques along with GID3 to deal with the noise problem. Both of these systems will be briefly outlined later in the paper.

## 2 Inducing Rules from Training Examples

Assume that there are $m$ attributes as described above. Let each attribute $A_i$ take on one of $r_i$ values $\{a_{i1}, \ldots, a_{ir_i}\}$. Assuming that on average an attribute takes on one of $r$ values, there

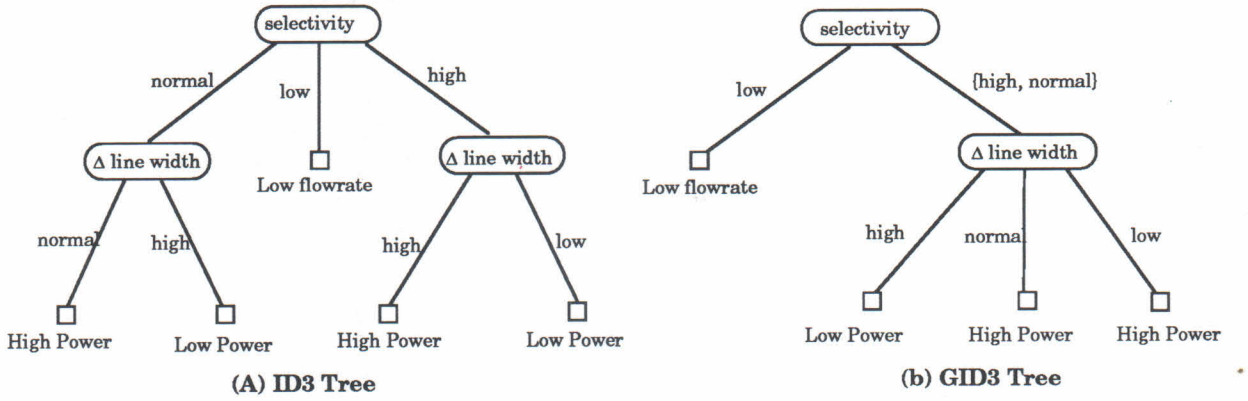


Figure 1: Decision Trees Generated for the Simple Data Set of Table 1.

are $p \cdot (r+1)^m$ possible rules for predicting the $p$ classes. It is computationally infeasible for a program to explore the space of all possible classification rules. In general, the problem of determining the minimal set of rules that cover a training set is NP-hard. It is therefore likely that a heuristic solution to the problem is the only feasible one. A particularly efficient method for extracting rules from data is to generate a decision tree [Brie84, Quin86]. A decision tree consists of nodes that are tests on the attributes. The outgoing branches of a node correspond to all the possible outcomes of the test at the node. The examples at a node in the tree are thus *partitioned* along the branches and each child node gets its corresponding subset of examples. A popular algorithm for generating decision trees is Quinlan's ID3 [Quin86], now commercially available.

ID3 starts by placing all the training examples at the root node of the tree. An attribute is then chosen to partition the data. For each value of the chosen attribute, a branch is created and the corresponding subset of examples that have the attribute value specified by the branch are moved to the newly created child node. The algorithm is then applied recursively to each child node until either all examples at a node are of one class, or all the examples at that node have the same values for all the attributes. An example decision tree generated by ID3 for the sample data set given in Table 1 is shown in Figure 1(a).

Every leaf in the decision tree represents a classification rule. The path from the root of

the tree to a leaf determines the conditions of the corresponding rule. The class at the leaf represents the rule's action.

Note that the critical decision in such a top-down decision tree generation algorithm is the choice of attribute at a node. The attribute selection is based on minimizing an information entropy measure applied to the examples at a node. The measure favors attributes that result in partitioning the data into subsets that have low class entropy. A subset of data has low class entropy when the majority of examples in it belong to a single class. The algorithm basically chooses the attribute that provides the locally maximum degree of discrimination between classes.

# 3 Overcoming Problems with ID3 Trees

It is beyond the scope of this paper to discuss the details of the ID3 algorithm and the criterion used to select the next attribute to branch on. The criterion for choosing the attribute clearly determines whether a "good" or "bad" tree is generated by the algorithm. Since making the optimal choice of attribute is computationally infeasible, ID3 utilizes a heuristic criterion which favors the attribute that results in the partition having the least information entropy with respect to the classes. This is generally a good criterion and often results in relatively good choices. However, there are weaknesses inherent in the ID3 algorithm that are due mainly to the fact that it creates a branch for each value of the attribute chosen for branching.

## 3.1 Problems with ID3 Trees

Let an attribute $A$, with possible values $\{a_1, a_2, \ldots, a_r\}$ be chosen for branching. ID3 will partition the data along $r$ branches each representing one of the values of $A$. However, it might be the case that only values $a_1$ and $a_2$ are of relevance to the classification task while the rest of the values may not have any special predictive value for the classes. These extra branches are harmful in three ways:

1. **They result in rules that are overspecialized**. The leaf nodes that are the descendants of the nodes created by the extraneous branches will be conditioned on particular irrelevant

attribute values. Since each leaf node corresponds to a classification rule, the irrelevant conditions will appear in the preconditions of the corresponding rules.

2. **They unnecessarily partition the data**, thus reducing the number of examples at each child node. The subsequent attribute choices made at such child nodes will be based on an unjustifiably reduced subset of data. The quality of such choices is thus unnecessarily reduced.

3. **They increase the likelihood of occurence of the** *missing branches problem.* This problem occurs because not every possible combination of attribute values is present in the examples.

The third problem can be illustrated in the ID3 tree shown in Figure 1(a). Consider two possible unclassified examples which are to be classified by the tree of Figure 1(a):

**ex1:** (Selectivity = low) & ($\Delta$ line width = low)

**ex2:** (Selectivity = normal) & ($\Delta$ line width = low)

Both ex1 and ex2 are examples that have combinations of attribute values that did not appear in the training set of Table 1. However, the tree readily classifies ex1 as being the result of an etch where the *flow rate was low*, but ex2 fails to be classified by the tree. This is because the subtree under the normal selectivity branch has no branch for *low* $\Delta$*line width.* We shall shortly illustrate how the occurence of *missing branches* may be avoided.

The main problem with the tree of Figure 1(a) is that the normal and high selectivity branches should not be separated. Low selectivity is the only value of relevance to the occurence of a problem. It would be desirable if the learning algorithm could somehow take account of such situations by avoiding branching on attribute values that are not individually relevant. This would reduce the occurence of the three problems listed above.

## 3.2 The GID3 Algorithm

To avoid some of the problems described above, we developed the GID3 algorithm[1]. We generalized the ID3 algorithm so that it does not necessarily branch on each value of the chosen

[1]The name GID3 stands for Generalized ID3.

attribute. GID3 can branch on arbitrary individual values of an attribute and "lump" the rest of the values in a single *default branch*. Unlike the other branches of the tree which represent a single value, the default branch represents a subset of values of an attribute. Unnecessary subdivision of the data may thus be reduced. Figure 1(b) shows the tree GID3 would generate for the data set of Table 1. Note that both examples, ex1 and ex2, are classifiable by this tree. The missing branch problem that prevented the tree of Figure 1(a) from classifying ex2 does not occur in the tree of Figure 1(b).

We now turn our attention to the application of GID3 to problems in the domain of semiconductor manufacturing. Interested readers are referred to [Fayy91, Chen88] for detailed accounts of the ID3 and GID3 algorithms, the attribute selection criterion, the weaknesses of the ID3 approach, and various performance measures to evaluate the quality of the resulting trees. Performance measures include error rate in classifying new examples and measures of the size complexity of the generated tree.

# 4 Applications of GID3 in Semiconductor Manufacturing

In this section we discuss several applications of the GID3 algorithm to semiconductor manufacturing domains. Most of these domains involve the reactive ion etching (RIE) process. The RIE process is a wafer etching process that promises increased precision and higher device density. It has been targeted for automation by the Semiconductor Research Corporation (SRC), a consortium of major U.S. companies in semiconductor manufacturing. One of the steps necessary for automation is the development of expert systems that determine process parameter settings based on given output constraints. The problem is that the process is not well-understood and no satisfactory methods for determining proper control settings exist.

To illustrate the types of industrial tasks to which GID3 can be applied, we describe application tasks from three categories: RIE process diagnosis, RIE process optimization, and an emitter piloting application. The goal of process diagnosis is to derive rules for diagnosing faults by deciding which process parameters are not correctly set. We describe two projects in this category. In the process optimization category, the project targeted deriving rules for

| power1 | power2 | power3 | time1 | time2 | time3 | wafer# | topology | ... | outcome |
|---|---|---|---|---|---|---|---|---|---|
| 1117 | 1134 | 490 | 8.98 | 7.20 | 5.0 | 18 | v | ... | erosion |
| 895 | 1139 | 492 | 10.26 | 8.2 | 5.0 | 24 | v | ... | erosion |
| 833 | 854 | 442 | 7.4 | 6.0 | 5.0 | 1 | v | ... | sleeves |
| 835 | 818 | 491 | 9.7 | 7.76 | 5.0 | 5 | v | ... | none |
| 859 | 835 | 490 | 9.9 | 7.92 | 5.0 | 2 | v | ... | none |
| 867 | 886 | 466 | 9.9 | 7.9 | 5.0 | 7 | v | ... | sleeves |
| 847 | 871 | 473 | 9.8 | 7.8 | 5.0 | 8 | v | ... | sleeves |
| 776 | 833 | 500 | 8.6 | 6.9 | 5.0 | 8 | v | ... | none |
| 771 | 813 | 490 | 8.7 | 7.0 | 5.0 | 4 | v | ... | none |
| 847 | 825 | 491 | 9.20 | 7.36 | 5.0 | 13 | v | ... | erosion |
| 851 | 843 | 490 | 17.2 | 4.3 | 5.0 | 6 | v | ... | none |
| 806 | 896 | 493 | 10.5 | 8.40 | 5.0 | 2 | v | ... | none |
| 844 | 822 | 493 | 9.16 | 7.33 | 5.0 | 14 | v | ... | erosion |
| 860 | 809 | 490 | 9.8 | 7.84 | 5.0 | 2 | v | ... | none |
| 867 | 825 | 452 | 9.7 | 7.76 | 5.0 | 2 | v | ... | sleeves |
| 878 | 819 | 454 | 24.1 | 14.4 | 5.0 | 1 | t | ... | none |
| 848 | 816 | 455 | 321.0 | 16.8 | 5. | 9 | t | ... | none |
| 806 | 778 | 484 | 22.5 | 9.0 | 5.0 | 8 | t | ... | sleeves |
| 881 | 795 | 467 | 22.7 | 13.6 | 5.0 | 1 | t | ... | none |
| 772 | 868 | 490 | 8.7 | 7.0 | 5.0 | 7 | v | ... | none |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 842 | 826 | 494 | 17.0 | 13.6 | 5.0 | 1 | v | ... | none |

Table 2: A Partial list of Data Logs for a RIE Process

dealing with situations where the operating point drifts away from the optimal operating point in the parameter space. Finally, we briefly discuss our effort on a project aimed at facilitating the knowledge acquisition effort for the development of an emitter piloting advisory expert system.

## 4.1 Process Diagnosis

The first project's goal is to acquire a set of RIE process diagnostic rules from a collection of production log data that contain fault inspection results by process engineers. The goal is to etch a specified pattern in the metal on a wafer.

Each data log contains about 60 data entries, including machine type, device specification, material and resist thickness, plasma time, power, DC bias, chamber pressure, gas flow, temperature, valve position and number of wafers. Since this is a multi-stage (three stage) etch, each stage has its own set of measurements. A distinct table slot is used for visual inspection after the etch. For this project, three types of inspection results are commonplace, namely, *normal, PR erosion,* and *sleeves*. A partial list of such data logs is shown in Table 2. Process

fault diagnosis has been a regular demand on process engineers. Whenever a fault is detected, its immediate cause needs to be identified and a decision is then made to correct the problem by adjusting process parameters directly or indirectly. Determining such adjustments is a non-trivial task. Abstracting these daily routines into rules that cover the task has been found to be difficult. Extracting general rules that can be transferred across different processes and be used to guide further reasoning to find physical laws governing the observed phenomena has been especially difficult. For this project, the difficulty in diagnosing the faults is due to the large number of process parameters, most of which vary greatly. These conditions naturally make the domain appear promising for the application of machine learning techniques.

Once the attributes and classes are determined, GID3 can be used to induce a decision tree from the available data. The decision tree is then transformed into a set of diagnostic rules because the rule format is more general and is easier for process engineers to comprehend. In order to get a set of rules which are general and reliable, and to overcome problems with noisy data, we used the RIST package described later. RIST runs GID3 many times. Each time, a random subset of the given data set is selected to induce rules. The rules are then tested against the whole data set. A set of statistical criteria are used to select a subset of the rules obtained from each GID3 session. We describe this method further in section 5.1.

Each of the rules generated provides a range of operating conditions with a prediction of the process outcome under those conditions. The power of our machine learning approach is manifested by the fact that most rules can predict the process outcome correctly with only one to three tests on the attributes. Figure 2, shows four of the rules generated for the data of Table 2. These rules exemplify pieces of compact, previously unknown, knowledge that are found useful by both process and knowledge engineers.

The second project was aimed at identifying relationships between RIE process problems, such as reduction in yield, and corresponding process parameters including the flow rate of each gas component and the chamber pressure for different etching steps. This data set was derived by regression analysis which statistically identifies the geometric patterns such as length, corners and gaps responsible for the yield loss. Classes consisted of dominating defect patterns. The details of this project with Westinghouse are discussed in [Frie89]. The rules derived by GID3

**Rule 1:** IF erosion is observed
THEN probably too many wafers are loaded;
Reduce number of wafers to $< 11$.

**Rule 2:** IF erosion is observed
THEN power at etch stage II may be too high;
Reduce power at etch stage II to $< 956.5$V.

**Rule 3:** IF sleeves are observed
THEN etching at stage I may not be sufficient;
Increase etch time for stage I to $> 8.15$ minutes.

**Rule 4:** IF sleeves are observed
AND stage III power $\in [451, 487]$
THEN Increase overetch percentage for stage II.

Figure 2: Rules for Diagnosing Faults of a RIE Process

were used to analyze and understand the process behavior.

## 4.2 Process Optimization

To achieve high yield, low defect density, and small geometry, RIE must operate at an optimal point in the input parameter space. The problem, known as process optimization, or process design, is the most important problem that a plasma engineer has to face.

The goal of process optimization is to find a set of input parameters, usually referred to as a *recipe*, such that the outputs can be optimized. For example, one may want to minimize the line width change under the condition that selectivity is higher than a certain value and uniformity is within a certain range.

A popular method to solve the process optimization problem is the Response Surface Methodology (RSM) [Berg82]. RSM is a statistical method in which the input parameters of RIE are treated as independent variables and the output parameters as response variables. For each response variable, multiple regression analysis is applied to generate a response surface to fit the data. Optimization techniques are then used to find a point that optimizes the response variables under the given constraint [Berg82]. However, RSM has its drawbacks. It is a static method in the following sense. Given a set of experimental data, a set of response surfaces can be generated and a fixed optimal point can be found. According to this optimal point, a recipe is formulated. The problem is that under a fixed recipe, the process might not be optimal because certain hidden variables which were not considered in the design process influence the process later. In other words, the response surface may drift so that the operating point may no longer be optimal. Obviously, what we need is a set of rules which tell us where to move

| *No.* | *press.* (mTorr) | *flow* (sccm) | *% flow* (%) | $\Delta$ *CD* ($\mu$m) | *CD unif.* ($\mu$m) | *Oxide loss* ($\AA$) | *Oxide unif.* ($\AA$) |
|---|---|---|---|---|---|---|---|
| 1 | 300 | 150 | 25 | 0.05 | 0.05 | 368 | 216.0 |
| 2 | 500 | 150 | 25 | 0.25 | 0.05 | 316 | 43.5 |
| 3 | 300 | 300 | 25 | 0.18 | 0.40 | 407 | 162.0 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 20 | 400 | 225 | 38 | 0.10 | 0.05 | 220 | 38.0 |

Table 3: Experimental Data for Process Optimization

in the parameter space to achieve the optimal output under the given constraints. This makes for a dynamic, rather than static, solution to the optimization problem.

The above analysis lead us to consider using machine learning to process optimization problems for RIE. Our objective was to generate a set of rules that tell us which input parameters should be changed, and in which direction, if the outputs are not optimal or do not satisfy the constraints. We discuss the general methodology that we applied to a particular project. Table 3 shows a partial set of experimental data we aquired from industry for this project. *Pressure, flow,* and *percent flow* are input parameters while the other five columns of Table 3 represent outputs. Each of the 20 items represents an experiment. To generate more data items, we used KARSM (described later in section 5.2) to obtain an excess of 500 examples from these 20.

To apply GID3, obviously, we can regard input parameters as attributes and output parameters as outcomes. Since GID3 requires that each example have one class (outcome), we first discretize and combine the output parameters. For the constraint variables, the values can be discretized as either *low* or *high* (based on the constraints). For the optimization variables, the values are discretized as different levels such as *very high, high, medium, low, very low*. The data is then fed into GID3 and a decision tree is obtained. In the decision tree, a leaf represents a class of outputs. A leaf may represent an optimal class, for instance, $\langle$*very low, high, low, low*$\rangle$ for "very low line width change, high selectivity, and low CD and Oxide uniformity". This is an optimal class when one wants to minimize the line width change under the condition that selectivity is higher than a certain value and uniformities are lower than certain values. Other leaves may represent faulty classes. For instance: "high line width change, low

selectivity, low CD uniformity, and high Oxide uniformity". To derive the required rules, the condition—the path from tree root to leaf—of a faulty leaf is compared to the condition of a leaf having an optimal class. The differences represent the changes needed to bring a faulty operation to an optimal one. For example, assume that the condition of the optimal leaf is "pressure below 300 mTorr, total flow above 180 sccm, percent flow above 40 %" and that the condition of a faulty leaf is "pressure above 312 mTorr, total flow from 215 to 300 sccm, percent flow below 37.5 %". We can now derive a rule

> *"if selectivity is lower than normal, Oxide uniformity is higher than normal, line width change is high, to minimize line width change, pressure should be decreased and percent flow should be increased".*

## 4.3 An Emitter Piloting Expert System Application

GID3 has also been applied successfully to acquire knowledge for minimizing steps in emitter piloting dispositions. Emitter piloting is a process of tuning integrated circuits printed in wafer so that device specification can be satisfied. This task is typically carried out by a human operator. The initial cycle time is determined by experience and cycle time adjustment is then guided by the measurement of two device parameters. If the values of the two parameters fall in their respective desired ranges, the cycle time is accepted for batch tuning and is called the "shooting time". Otherwise, it is either increased or decreased to bring parameter values within desirable ranges. The number of steps needed before success in such a process is greatly affected by operator experience. Such experience is very valuable but is very difficult to encode in rules. The purpose of the project was to collect all sequences of trials conducted and to use the machine learning approach to attempt to extract the knowledge underlying the actions of human operators. See [Yang89] for further details of this domain.

The raw data used in knowledge acquisition is composed of numerous experiment data logs, each of which consists of sequences of cycle time adjustments targeting one shooting time. For every trial in each sequence, the cycle time used and two parameter measurement values were recorded. GID3 was used to learn rules for jumping to a shooting time from an arbitrary cycle time by letting each data point be the condition under which certain adjustments can be made to achieve a certain shooting time. The difference between the current cycle time

and the actual shooting time for each example was taken to be the predetermined class. The rules induced by GID3 were evaluated by an expert and were deemed satisfactory. In this project GID3 was used as a knowledge acquisition tool to gather rules for incorporation into an expert system developed by Harris Semiconductor.

# 5 Dealing with Industrial Data Problems

In conclusion to this presentation, we focus on two special problems encountered in industrial applications and the solutions we devised to combat them. The problems are:

**Noisy Data:** attribute values may be erroneous due to human recording errors, imperfect sensor repeatabilty, or defects in process equipment or sensors.

**Limited Training Data:** the training data may be small in size and conducting more experiments may be too costly.

## 5.1 Dealing with Noisy Data

Empirical learning algorithms are typically sensitive to the presence of noise because they rely solely on data to discover rules. Typically, they are not intended to have access to special domain knowledge to guide their decisions. Noise in a training data set may cause irrelevant rule conditions to be selected. The solution we devised combats noise in two ways:

1. Statistical evaluation (pruning) to identify and remove irrelevant conditions from rules.
2. Random sampling of multiple training sets and selection of statistically significant rules from the trees generated for these training sets.

After a decision tree is generated, a statistical test is applied to remove rule preconditions that are deemed statistically irrelevant, resulting in more general, *pruned*, rules. The statistical test we use is Fisher's Exact Test. It measures the probability that a hypotheses (in this case a condition of a rule) is irrelevant to the outcome. If this probability is higher than a small value, the condition is discarded. For details of the statistical test see [Quin87, Finn63].

This method of statistical pruning deals with the rules produced from one run of GID3. By randomly sampling subsets from a given training set, many trees can be generated. For each tree, we apply the statistical test and keep only the "good" rules. Finally, a subset of the

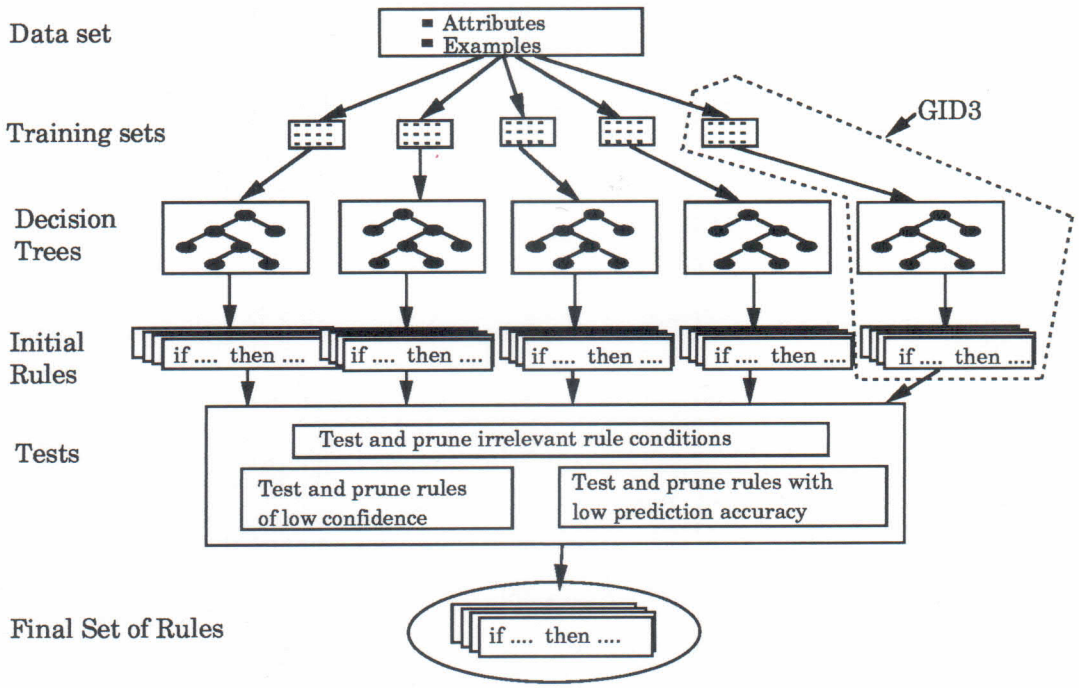


Figure 3: Data Flow Diagram for RIST

surviving rules that covers the original training set is selected. When coupled with a method for statistical significance testing, the multiple random sampling of the training set has proven to be an effective technique for extracting a compact and reliable set of rules from the original training set. The method, illustrated in Figure 3, has been implemented in a software package named RIST (Rule Induction and Statistical Testing).

## 5.2 Dealing with Limited Training Sets

The learning algorithms we discussed do not use any special domain knowledge about the data during tree or rule generation. They rely on the availability of large training samples to detect the presence of meaningful reliable patterns or correlations. In some cases however, obtaining training examples may be an expensive process. In this case, only a limited training set may be available.

In section 4.2 we mentioned that RSM is standard methodology used in process optimiza-

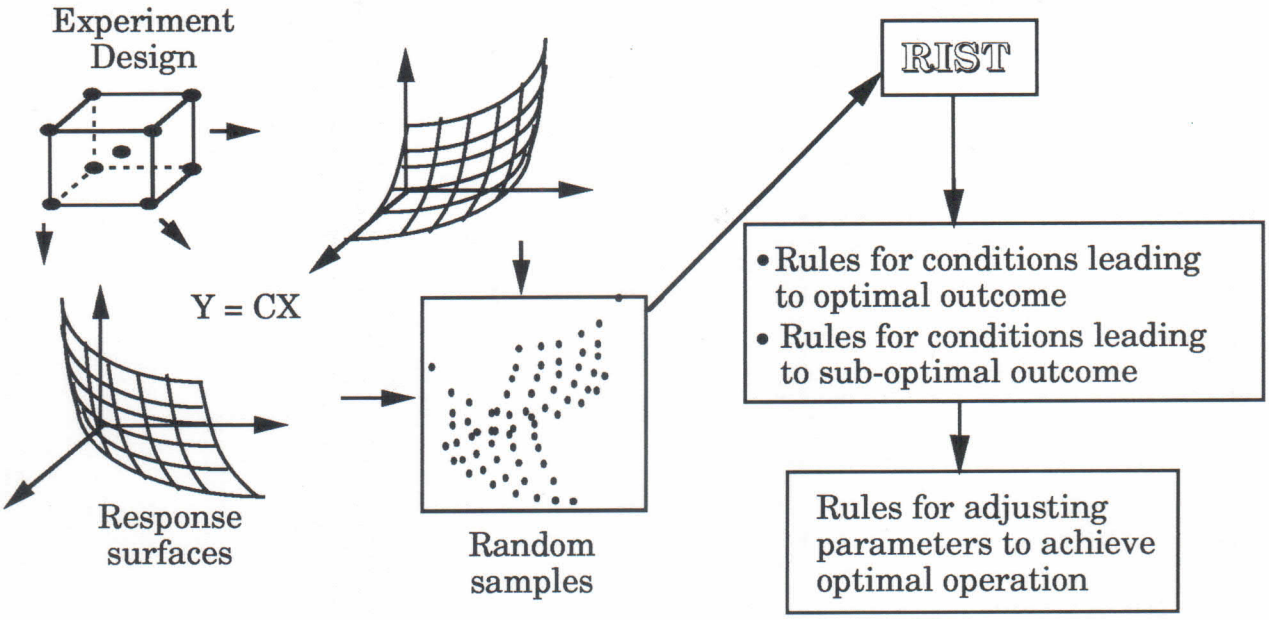


Figure 4: Data flow diagram for KARSM

tion. The core component of this methodology is to approximate the given data by a polynomial surface. Once the data is fitted with a surface, each point on the surface corresponds to a specification of the input and output variables. Since we know the target conditions that the output variable must satisfy, we can quantize the output value into: { *Good, Bad*}, or into a finer partition such as { *Very Low, Low, Good, High, Very High*}.

At this point, we can randomly choose a combination of input values, and the response surface will give us the associated discretized output. As is shown in Table 4, 500 examples can be extracted from the original data of Table 3. This gives us a method of extracting an arbitrarily large training sample from an original small training set. Of course, the "usefulness" of the extracted large training set depends on the correctness of the response surface generated to fit the original data. The generated data set is then fed into RIST (or GID3) to extract qualitative rules that describe the behavior of the data.

We have implemented this procedure in a software package called KARSM (Knowledge Acquisition from Response Surface Methodology). It is Illustrated in Figure 4.

The reader may wonder why we do not stop with the response surface and use it directly to describe the process. The answer is that the response surface does not give an easily *un-*

| *No.* | *pressure* | *total flow* | *percent $Cl_2$ flow* | *class* |
|---|---|---|---|---|
| 1 | 377 | 192 | 49 | ab,l,l,l |
| 2 | 342 | 297 | 36 | h,l,l,h |
| 3 | 491 | 241 | 34 | ab,h,l,h |
| ... | ... | ... | ... | ... |
| 499 | 452 | 211 | 37 | m,h,h,l |
| 500 | 303 | 185 | 42 | vl,h,l,l |

Table 4: Classified Random Samples

*derstandable* description of the process. By generating rules from the surface we essentially extract a *qualitative* description of the behavior of the process in terms of desirable and undesirable regions of the output space. The procedure for extracting, from the decision tree, the qualitative rules that specify the direction in which process parameters should be changed, when the operating point drifts, is described in section 4.2.

# 6 Conclusion

We have introduced the general problem of extracting rules from data for the purpose of automating the knowledge acquisition process. We described the GID3 algorithm for inducing decision trees and its use in automating knowledge acquisition in several application domains. Given a training set of data, the algorithm produces a decision tree for predicting the outcome of future experiments under various, more general conditions. The tree may then be translated into a set of rules for use in expert systems. We also introduced two extensions of GID3 to deal with noise and limited training set availability.

We have utilized GID3 to extract knowledge from data in several application domains. In each case, the decision trees obtained, or the English version of the generated rules, were sent to the engineer who supervises the experiments and provides the data. The derived rules were judged to be consistent with the data and conformant with the engineers' expectations. A derived model, or a discovered pattern, that is consistent with a process engineer's expectation is of great value in two ways:

1. It provides a previously unavailable mechanical means for classifying events or relating faults to parameters.

2. It gives the process engineer further insight into the process by making explicit a pattern that was previously implicit as part of the engineer's "intuition" about the process.

We believe that machine learning techniques have an important role to play in the automation and improvement of manufacturing techniques as well as many other diagnostic tasks. A machine learning approach avoids the ubiquitous "knowledge acquisition bottleneck" by minimizing the required interaction with domain experts and focussing on a resource that is much easier to obtain: experimental data. A further advantage of using an induction system such as GID3 is that knowledge acquisition can be fast, accurate, and most importantly, can be automated.

## Acknowledgements

This work was supported by the University of Michigan SRC Research Program under contract #89-MC-085. We would also like to thank Hughes Microelectronics Center for their partial support of this work in the form of an unrestricted grant.

## References

[Berg82] Bergendahl, A.S., Bergeron, S.F., and Harmon, D.L. (1982). "Optimization of plasma processing for silicone-gate FET manufacturing applications." *IBM Journal of Res. Dev. vol. 26*, no. 5.

[Brie84] Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J. (1984). *Classification and Regression Trees*. Monterey, CA: Wadsworth & Brooks.

[Chen88] Cheng, J., Fayyad, U.M., Irani, K.B., and Qian, Z. (1988). "Improved decision trees: A generalized version of ID3." *Proceedings of the Fifth International Conference on Machine Learning*. pp. 100-108. Ann Arbor, MI.

[Fayy91] Fayyad, U.M. (1991). *On the Induction of Decision Trees for Multiple Concept Learning*. Dissertation in preparation, EECS Dept., The University of Michigan, Ann Arbor.

[Fieg81] Fiegenbaum, E.A. (1981). "Expert systems in the 1980s." In Bond, A. (Ed.), *State of The Art Report on Machine Intelligence*. Maidenhead: Pergamon-Infotech.

[Finn63] Finney, D.J., Latscha, R., Bennett, B.M., and Hsu, P. (1963). *Tables for Testing Significance in a 2x2 Contingency Table*. Cambridge: Cambridge University Press.

[Frie89] Friedhoff, C.B., Cresswell, M.W., Lowry, C.R., and Irani, K.B. (1989). "Analysis of intra-level isolation test structure data by multiple regression to facilitate rule identification for diagnostic expert systems." *Proceedings of the International Conference on Microelectronic Test Structures*. Edinburgh, Scotland.

[Quin86] Quinlan, J.R. (1986). "Induction of decision trees." *Machine Learning 1, No. 1.* pp. 81-106.

[Quin87] Quinlan, J. R. (1987). "Generating Production Rules From Decision Trees." *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*. Milan, Italy.

[Yang89] Yang, Y.K. (1989). "EPAS: An emitter piloting advisory expert system for IC emitter disposition." *Proceedings of Semicon West.*