Machine Learning 1: 177–226, 1986

A General Framework for Induction and a Study of Selective Induction

LARRY RENDELL (RENDELL @B.CS.UIUC.EDU)

Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, U.S.A.

(Received December 15, 1985)

Key words: induction, uncertain and incremental learning, concept formation

Abstract. This paper has two major parts. The first is an extensive analysis of the problem of induction, and the second part is a detailed study of selective induction. Throughout the paper we integrate a number of notions, mainly from artificial intelligence, but also from pattern recognition and cognitive psychology. The result is a synthetic view which exploits uncertainty, task-guidance, and biases such as language restriction. Some of the main themes and contributions are as follows. (1) Practical induction is really a problem of efficacy and efficiency (power). (2) Search in a space of hypothetical concepts is governed by a *credibility function* which combines various knowledge sources in a single subjective probability or belief measure μ. (3) The amount of knowledge supplied by various sources can often be quantified; these sources include various biases and the learning system itself. (4) Induction is equivalent to discovery of a *utility function* u, which captures the purpose or goal of induction. (5) The difficulty of induction may be characterized by the form of u. Smooth or coherent functions mean selective induction, which has had the most attention in machine learning. (6) Systems for selective induction are more similar than commonly understood. By juxtaposing them we can discover similarities and improvements. (7) Our analysis suggests a number of incipient principles for powerful induction.

0. Introduction

0.1 The prevalence and nature of induction

Induction is an important but complex problem which has been extensively studied in psychology, philosophy, pattern recognition and artificial intelligence. Increasing interaction among these four fields has promoted new perspectives, and redefined kernals from each discipline may soon provide a more complete understanding of induction, along with improved mechanization. This paper presents some recent

findings and approaches, primarily from AI, but to some extent from the other three fields. We attempt to integrate several aspects and methods.

The problem of induction is complex. It can be considered as the compression of massive data, as the formation of meaningful concepts, or as the discovery of coherent descriptions. Induction may also be thought of as the imposition of order, or as the expression of invariance. Induction presumes some purpose or abstract goal. Devoid of purpose, induction is generalization, the formation of subsets or classes from a universe of patterns, events, or objects. An object might be a visual grid, the state of a checker board, a patient with a disease, or countless other items of interest. But here we return to purpose; objects within a class are similar with respect to some goal. The classes are cohesive categories described as purposeful concepts. Angluin & Smith (1983), Christensen (1964), and Watanabe (1969) provide many views of induction.

Since induction reduces the number of categories to manage, it promotes economy of space and time. Because a concept description embodies not only observed objects, but also similar objects yet to be encountered, induction is predictive. Efficient and accurate prediction is one of the main characteristics of intelligence.

0.2 Themes and purposes of this paper

This paper has a number of themes. First, it examines representations, methods and principles for making induction effective yet computationally tractable. Both fundamental and advanced issues are considered, often in a new light, as some novel schemes are unified with some better known ones. We examine and synthesize aspects such as inductive bias, goal-direction, uncertainty, and power.

A related goal of this paper is to unify, to draw parallels between some learning systems which seem more diverse than closer examination reveals. The systems we consider tend to be the ones which are better known, better evidenced, or seminal. Some issues are clarified, some approaches are juxtaposed and synthesized, and some ideas are more fully articulated and developed than previously.

Another major purpose of this paper is to present a methodology for studying practical induction. We want to understand the reasons for the success of various learning systems, and the degree to which various components or methods affect their power. To do this we need some concrete ways of comparing systems and their components. If standards of comparison and measures of performance can be developed, they can help to identify underlying principles, pinpoint useful mechanisms, and consequently sharpen our investigations. Measures of performance may even permit greater mechanization.

The scope of our examination is necessarily limited. We primarily consider systems for *supervised* learning (learning from examples), although our approach suggests a certain similarity between this and unsupervised learning. We cover many aspects of

supervised learning, including uncertainty and incremental revision. However, after presenting a basic framework for the general problem of induction, this paper examines a relatively straightforward kind of induction, which is sometimes called "selective" (Michalski, 1983).

Apart from these restrictions, our examination is rather extensive. In the following major part of the paper we shall develop a framework for analyzing and understanding induction. This will include quantitative definitions of important measures, such as the credibility of an hypothesis and the amount of knowledge induced. In Part 2, we shall use our framework to analyze selective induction, to synthesize approaches to it, and to extract incipient principles.

1. A view of practical induction

This part of the paper examines a number of fundamental issues in practical induction and introduces some terms and concepts which will be used later. The first section explains what induction is and details why heuristic methods are required to discover good hypotheses. Section 1.2 examines means for assessing alternative hypotheses in terms of their underlying purpose or "credibility"; this is the issue of efficacy. The third section characterizes ways of discovering credible hypotheses quickly, for example by imposing "inductive bias"; this is the efficiency concern. Finally, Section 1.4 presents a uniform measure for several aspects of induction, including imposed bias, problem difficulty, and acquired knowledge. The resulting framework will unify methods and may encourage discovery of principles for learning.

1.1 What is induction?

Induction may be used in many ways, e.g. to optimize control for tasks (Rendell, 1983a), to learn rules in expert systems (Michalski, 1983), or to discover concepts for any purpose. But underlying any domain application or learning approach is the same fundamental problem. In its simplest form, induction is the partitioning of a set of objects, patterns, or events into subsets, i.e. induction is class formation. In contrast to this colorless view, however, we normally think of induction as having a rich meaning. For one thing, construction of new objects or descriptions may be required, and even without this complication, something more than just class formation is essential. There is no reason to choose one classification over any other unless some preference criterion is imposed; when induction takes place, similar objects are compressed into classes which are *coherent* categories or *meaningful* concepts. See Watanabe's (1969) "theorem of the ugly duckling", Mitchell's (1980) discussion of "bias", and Murphy's & Medin's (1985) "conceptual coherence".

1.1.1 Fundamental definitions and perspectives

We shall employ a dual approach to induction which is somewhat unusual in AI. Instead of restricting our investigation to logic or some other representation language, we sometimes take a more abstract standpoint independent of language. This will allow us to clarify certain issues such as the amount of learning an inductive system is doing. At other times we shall focus on representation and description, and this perspective will allow examination of agents such as inductive operators and learning systems. The cumulative effect of our combined study will be to provide a basis for comparing methods, and to identify principles of inductive power. To begin, we need to define some terms.

Definition 1. Objects and attributes

A pattern, event, or *object* can be any kind of entity, although for our purposes an object is actually a description, since an entity must be represented. Hence a desk becomes a list of its properties, a visual scene becomes a matrix of pixels, and a game position becomes a state. Instead of a physical entity, an object can be any construct whatsoever.

An object can be described at any level of abstraction, depending on its *attributes*, which are ascribed properties, or variables taking on values in a specified range. For example, a checker board might be expressed using features such as piece advantage, mobility, etc. Instead the board could be described in terms of the contents of its individual squares. Or the board might even be a visual grid of pixels with game squares undiscriminated. In other words, what we consider an object in one problem might be a whole collection of objects in a different context. Attributes can be abstract *features*, like piece advantage, or they may be elementary *primitives*, like checkerboard contents.

An object can also be structured, but structure can be hidden in a description (e.g. consider the attribute "mobility" in checkers). In this paper an object is either its name, e.g. o_1, or an object is a vector of attribute values $x = (x_1, x_2, \ldots, x_k)$.

Definition 2. Classes and concepts

Given a set or universe of objects, a *class* is a subset of these objects. The term *concept* is usually meant as a description of a class (its intension), although the distinction between "class" and "concept" is not always sharp, and the two words are synonymous for some purposes. The two terms are similar when we consider the more abstract properties and consequences of induction, such as class formation irrespective of method. In other cases, especially when we discuss implementation

details of induction such as representation and search, "concept" not only means a description of a class, but it also connotes purpose, regularity, and structure.

Definition 3. Induction

Basically, induction is class formation, the partitioning of a universe into subsets. The simplest kind of induction classifies only the original universe of objects (cf. the "single representation trick", Dietterich, 1982). In some cases, the classes are partially formed, and the problem is to complete the process. Our definition stands, however, because the initial aggregates can be considered as objects: an object may be a whole collection of less abstract objects.

Another possible complication is that objects may require construction from subobjects (this means the formation of relations). But here, too, there is a universe of objects to be classified, a universe of relations (the relations are subsets of cartesian products having subobject components). However, when a system must learn structuring, the inductive process is more difficult. Although we shall use some complex representation in this paper, we shall be concerned primarily with problems which do not require extensive restructuring of the original descriptions. (This is the topic of a subsequent paper.).

A meaningful definition of induction presumes some purpose or goal, so we reserve the term *induction* to mean goal-oriented class formation. Further, induction suggests testing relative to the goal. The terms "concept" and "induction" are strongly related.

Definition 4. Hypotheses

An *hypothesis* is fundamentally a partition of the universe, i.e. hypothesis formation is class formation (see Table 1). In more meaningful terms, an hypothesis is a purposeful assertion about objects in the universe, a product of induction before testing has been completed. While an hypothesis could be an assertion involving any number of classes, we can always consider one dichotomy at a time. For example, a universe of visual grids could be categorized into 26 letters plus a nonsense class, but instead each letter can be learned separately by performing induction 26 times. In this paper an hypothesis is usually a dichotomy or a description of a dichotomy. Hence an hypothesis is frequently a candidate concept.

Definition 5. Practical induction and system power

Learning systems manipulate descriptions of classes, or hypotheses. A candidate concept H is normally associated with some purpose or goal, and the extent to which H satisfies the purpose determines the quality of the induction. Since we want learning systems to solve problems quickly, we could say that *practical induction* is the study and application of effective and efficient concept learning, using methods having significant scope. Scope means efficacy and efficiency across domains. Together, efficacy, efficiency and scope determine inductive *power.*

Our ultimate goal is to understand and to construct algorithms for practical induction. By "inductive algorithm" or "induction method", we mean an algorithm which forms concepts of high quality, but not necessarily the optimal one (cf. Angluin and Smith, 1983).

1.1.2 Illustrative examples of class formation

To perform practical induction, we must consider notions about coherent categories and meaningful concepts; these involve ideas such as similarity and conceptualization. But let us also illustrate certain abstract properties of class formation, namely the combinatorics, which will underscore the great difficulty of the problem. The examples below will also suggest the ubiquity of induction, which appears in every domain from vision to problem solving. These examples will recur throughout the paper to illustrate various points.

Example 1: "Single concept" learning. This is the problem of creating just two classes: C and its complement $U - C$ relative to *universe* U (if we think of C as a concept description, we would write the complement of C as . C). There are two sets of objects involved in this problem. One is the universe of all N possible objects which might ever arise $\{o_1, o_2, \ldots, o_N\}$.[1] The other set is the *training set* of n objects ($n \leq N$). Each member of the training set is presented as a positive or negative example of C. This is *supervised learning* (Duda & Hart, 1973; Tou & Gonzalez, 1974) or *learning from examples* (Michalski, 1983; Winston, 1984). Whenever $n < N$ and C covers more objects than just its positive training instances, we have a case of induction (note the potential for prediction).

The number of different ways to dichotomize the universe is extremely large. To define a category, we may select each object independently of every other one, so there are 2^N possible dichotomies in all (counting C and $U - C$ twice, once for the concept, and once for its negation). This is the number of classes or subsets of the

[1] The universe could have an infinite number of elements, but in practice N is usually finite, because of the way objects are described. We shall elaborate shortly.

universe, as shown in Figure 1. If N is merely 20, the number of hypothetical classes or candidate concepts is over a million.

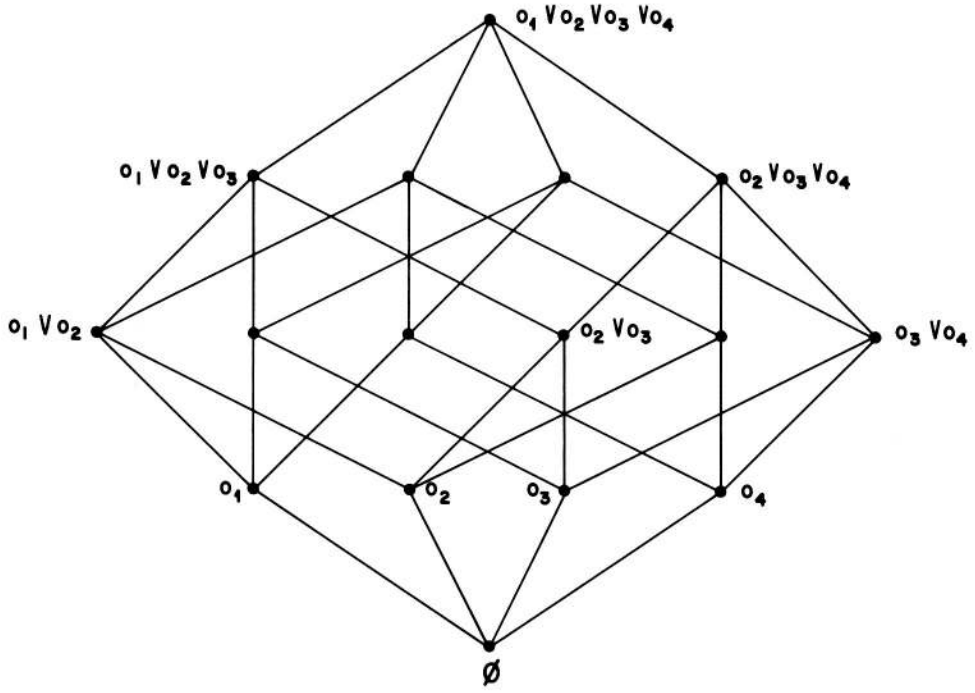

Figure 1. Propositional lattice or version space. The nodes are labeled with disjuncts of objects. This structure interrelates all possible concepts, and search for a credible hypothesis (a classification matching the purpose) may take place by moving up and down the lattice. Search would then be simplified, since confirmations and denials of one candidate support and preclude other candidates. E.g. rejection of $(o_3 \vee o_4)$ implies $(o_2 \vee o_3 \vee o_4)$ should be eliminated (note that the lattice facilitates this perfectly).

Example 2: Symbol recognition. Consider a small 10 × 10 grid of bits which encodes a symbol. Since every one of the 100 pixels can be on or off independently of the others, the number of different grids is $N = 2^{100} \simeq 10^{10}$. If each posible grid represents a letter of the alphabet, the number of different ways of forming the necessary 27 classes (one for each letter, plus a nonsense category) would be $\sim 27^N = 27^{[10^{10}]} = 10^{[1.4 \times 10^{10}]}$ (See Anderberg, 1973, p. 3.) i.e., a multiplication symbol.

This problem can be reduced to learning one symbol at a time. Instances of an "A" would first be given as positive examples, and all other instances would be negative. This would be repeated for each letter in turn (each time with appropriate positive and negative training sets). Hence the problem becomes 26 cases of single concept learning, each with $2^{[10^{10}]}$ hypotheses.

This immensity can be drastically improved if we replace the 100 primitive attributes (pixels) by a few abstract features which indicate the presence and position

of standard strokes and curves. For example we might define 6 features, say with 5 values each; then the number N of objects in the universe is $5^6 \simeq 15600$. Hence the number of hypotheses becomes $2^N = 2^{15600} \simeq 10^{4680}$; this, however, is still greater than the number of particles in the physical universe.

Example 3: State-space heuristic search. Consider the game of checkers. There are 32 squares which may be legally occupied, each in five possible ways (black king, black man, no piece, red man, or red king). If there were no constraints in number and placement of the pieces, the number of possible checkerboard configurations or states would be $5^{32} \simeq 2.3 \times 10^{22}$. Accounting for the various restrictions in placement and quantity, the actual number of states can be estimated to be roughly $N = 10^{18}$.

We might wish to know which of the 10^{18} states are wins, which are losses, and which are draws (assuming something consistent about our opponent, e.g., that she is a perfect player). The number of ways of classifying all of these board configurations into three categories (win, loss, or draw) is huge: $3^N = 3^{[10^{18}]} \simeq 10^{[4.8 \times 10^{17}]}$. Each one of these $10^{[4.8 \times 10^{17}]}$ is a different hypothesis, just one of which is correct.

In typical machine learning experiments with games, primitive board descriptions are mapped into abstract features such as piece advantage and center control. This improves the situation markedly, because the size of the universe is thereby reduced to roughly a million (Samuel, 1967). The number of hypotheses is now $3^{[10^6]} \simeq 10^{[4.8 \times 10^5]}$, still practically infinite.

Discussion. We can formulate several conclusions from these examples. One is that no matter what the domain, induction is fundamentally the same abstract problem. For example, inducing evaluation functions is much like single concept learning (Rendell, 1983b). Another conclusion is that induction is procedural, since hypotheses are well defined and enumerable. But perhaps the most important fact for practical induction is that algorithms already exist which do much better than our combinatorial analyses suggest. Why is this?

Part of the answer lies in a *criterion* for constructing and assessing hypothesis. The inductive criterion or "credibility" depends on two components (Watanabe, 1969). One is evidential, and relates to the intended purpose of the induction. The other component is extra-evidential, and depends on preconceived notions about desirability, such as simplicity of expression.

We shall discuss the crucial topic of credibility briefly now, and examine it later in detail. In Example 1 (single concept learning), and Example 2 (symbol recognition), one way to define the evidential component is obvious and direct: an hypothesis is credible if it classifies objects correctly. In Example 3 (checkers), a teacher could indicate the quality of an hypothesis, or else a program could measure performance autonomously. In contrast to these straightforward interpretations of

evidential credibility, the extra-evidential component is difficult to specify and not well understood. For Example 1, we might prefer "simple" concepts as in Figure 2, but simplicity is tricky. In Example 2, an "elegant" symbol description might involve intermediate concepts such as strokes and curves which would be uniform over all letters, but implementing elegance to create these intermediate concepts from pixels would be intricate.

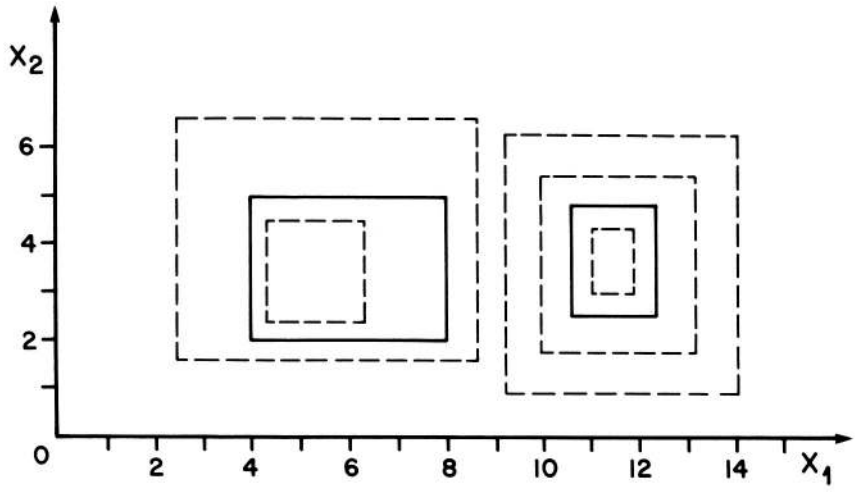

Figure 2. Concept descriptions biased by implicit disjunction. Here objects are points in feature space described by two attributes. Imagine a movable and deformable window, through which the visible objects define a concept. If the window is rectangular, the concept is conjunctive; for example the solid rectangle on the left is $(4 \leq x_1 \leq 8)$ & $(2 \leq x_2 \leq 5)$. This expression implicitly captures disjuncts of basic objects. Disjunctive concepts may be expressible as a single neighborhood, or they require more than one window. Limiting the number of windows favors conjunctive descriptions (of implicit disjunction).

Despite the difficulty of formulating some criteria, credibility may be used to assess hypotheses, to constrain their expression, and to guide their formation. We may now begin to distinguish between blind hypothesis generation and controlled search.

1.1.3 Simplistic versus practical induction

Finiteness and solvability. While the combinatorics are extreme in our illustrative examples, induction could nevertheless be performed simply by generation and testing. Given a universe of N objects $o_1, o_2, \ldots, o_N$, and a desired number of classes m, each of the roughly m^N partitions (hypotheses) could be formed, and then assessed according to some credibility criterion. Theoretically, induction is directly attainable, providing that N is finite. Even an infinite number of objects may not be a problem. We normally express objects in terms of a limited number k of attributes x_i $(1 \leq i$

$\leq$ k). If an attribute has an infinite range, it may usually be quantized into c values, so the universe of objects U is then compressed into $N \leq c^k$ descriptions.

Figure 1 illustrates the complete set of possible inductions when the universe contains only N = 4 objects. There are $2^4 = 16$ classes that can be formed here. This *propositional lattice* (Watanabe, 1969) is a simple *version space* (Michell, 1978). This particular diagram does not describe individual objects but only names them.

Advantage of intension. If we describe objects (Figure 2) instead of just naming them (Figure 1), we provide an alternative way of specifying classes. Rather than a subset C of the universe, we can specify a description H_c of C, expressed in terms of attributes x_i (this is definition by *intension*). Despite the apparent equivalence, there is a distinct advantage in specifying a description H_c instead of an amorphous subset C.[2] As shown in the "theorem of the ugly duckling" (Watanabe, 1969), no one class is inherently better than any other. There is absolutely no basis for induction unless the training examples, together with the inductive method, produce concepts expressing similarities to and extensions of the original examples. This can occur only if we impose some preference when forming a class, and any preference is fundamentally arbitrary. However, the illogic disappears if we impose a language in which to represent an hypotheses H_c. The language and our preferences for its use supply the inductive predisposition, which Mitchell (1980) has called a *bias.*

One bias is to prefer hypotheses described as windows or neighborhoods, as shown in Figure 2. A neighborhood may be expressed as a conjunction of attribute ranges $(a_1 \leq x_1 \leq b_1) \& (a_2 \leq x_2 \leq b_2) \& \ldots \& (a_k \leq x_k \leq b_k)$. This captures disjunctions, but in a restricted fashion. Suppose that a concept or hypotheses can only be a single neighborhood; in this case any disjuncts must be local. We may relax the bias somewhat by permitting a concept to be a few neighborhoods. This allows the expression of more hypotheses, but if the number of windows m is finite, we have restricted the set of possible concepts, and therefore we have imposed a bias.

If m is a small number, we call the bias *implicit disjunction.* Implicit disjunction has often been investigated in its specialized forms as discriminant analysis, internal disjunction, etc. (Dietterich, 1982; Duda & Hart, 1973; Michalski, 1983). This bias may occur in humans (Bruner, Goodnow & Austin, 1956). Like many other biases, implicit disjunction is a consequence both of the language (e.g., feature space rectangles) and of their preferred use (small numbers of them). This preference is not entirely arbitrary, though. It is cognitively economical (no storage is required) to increase the size of a neighborhood, since only the limits need to be altered in the conjunction of attributes $(a_1 \leq x_1 \leq b_1) \& (a_2 \leq x_2 \leq b_2) \& \ldots$ (cf. the minimization of logic circuits). In contrast, adding new windows requires more storage, unless the

[2] Using concepts instead of classes may also cause a problem. Depending on the description language, more than one concept may represent the same class. This difficulty is precluded in some approaches (Rendell, 1985b).

windows themselves are regularly arranged and *this* regularity can be summarized as another kind of implicit disjunction as in (Rendell, 1985b).

Instead of describing a concept using conjunctions of attribute ranges, we may represent the concept using its centroid or *prototype*. This is a variant of implicit disjunction having only one window with fuzzy boundaries; the degree of class membership becomes fainter with increasing distance from the prototype. The essence of implicit disjunction is that neighborhoods are meaningful, so like the strict logical case, a prototype description is also naturally depicted using a feature space diagram (Figure 2). Prototype representations have been studied in (Duda & Hart, 1973; Medin & Smith, 1984). We shall return to these ideas in Part 2.

Summary. To learn a concept in theory, we may simply enumerate and test hypotheses, but in practice we must use various aids. One is to impose structure on the set of hypotheses to exploit ordered search (e.g. Figure 1). Another aid is to prefer certain hypotheses and possibly ignore others (e.g. Figure 2). Notions of testing and bias bring us to one of our main themes: practical induction has three aspects, *efficacy, efficiency*, and *scope*. In the following sections we consider ways of judging and aiding efficacy and efficiency without compromising scope too much.

1.2 Efficacy: the quality of an induction

In our discussion of illustrative examples, we began to address efficacy through inductive purpose and hypothesis testing. Let us consider detailed methods.

1.2.1 Hypothesis assessment

We define the *assessment function* μ mapping the set $\{H\}$ of all hypotheses into some range R. (Recall that if the size of the universe is N, and we are learning single concepts, then there are 2^N hypotheses in all.) If we want a strict definition of μ, all but one of the members H of $\{H\}$ (i.e. $2^N - 1$ of them) would have $\mu(H) = 0$, while $\mu(H) = 1$ for just one $H_i \in \{H\}$. In this rigid view, the range R of μ is binary; see Table 1(a).

But is binary assessment function adequate? For small problems like Example 1 with $N = 4$, a program can afford to learn deterministically: training examples are usually noiseless, and search for a concept is inexpensive (Figure 1). However, when the problem is larger, there are difficulties. The most obvious is the exponential growth of hypothesis set $\{H\}$ with N. Because of the immense quantities involved, constraints are usually imposed on the language expressing a concept (as in Figure 2), and we can no longer be sure that any of the limited possibilities express the concept properly (see Figure 1c). Moreover, data tend to be sparse, and perhaps unreliable, so they offer little help. The doubts introduced by these problems and

compromises suggest a more flexible view of hypothesis assessment.

Binary assessment appears inadequate for yet another reason: there are gradations in the quality of an hypothesis. Consider again that induction is class formation and imagine the degree of error. In Example 1 (single concept learning), if the correct concept is ($o_2 \vee o_4$), then the hypothesis $H_1 = o_2$ is better than $H_2 = (o_1 \vee o_2 \vee o_3 \vee o_4)$. To see this, ignore the type of misclassification (omission versus commission) and count the number of errors. H_1 is wrong just once (about o_4), whereas H_2 is wrong twice (about o_1 and o_3). A more realistic case is that of Example 3 (checkers), where a few misclassifications (e.g. draws misrepresented as wins) would not matter much, since a relatively small number of incorrectly identified board configurations might never arise in a typical game.

Although just one hypothesis is correct, many others may be very good. Instead of a rigid binary judgment, then, a more flexible definition of our assessment function would be better. A rational choice for a graduated assessment function is $\mu(H)$ = the proportion of correct categorizations of objects. This is the probability that an object would be correctly identified by hypothesis H. Because of the large number of objects typically to be classified, there are many different $H \in \{H\}$ for

Table 1. Set of hypotheses and hypothesis assessment function μ. Although there are an immense number of them, hypotheses might be enumerated and each H evaluated by the assessment function μ, which gives a belief or credibility value. In (a), the range of values $R = \{0, 1\}$, signifying a binary decision. In (b), $R = [0, 1]$, indicating degree of belief we have in H. The latter is more flexible since it permits degrees of belief and facilitates incremental updating. In (c), the domain of μ is restricted, as only a fraction of the hypotheses are expressed (the representation language may constrain candidates).

Hypothesis	(a) Rigid, Binary μ	(b) Degree of Belief μ	(c) Restricted μ Domain
H_1	0	0.1	–
H_2	0	0.7	0.7
H_3	0	0.3	–
H_4	0	0.4	–
H_5	0	0.6	0.6
H_6	0	0.3	–
H_7	0	0.7	–
.	.	.	.
.	.	.	.
H_i	1	1.0	1.0
.			
.			

which μ(H) is very high, although these constitute just a small fraction of the total number of hypotheses in {H}; see Table 1(b).[3]

1.2.2 Representing belief

We have been considering the assessment μ(H) as the objective probability of correct classification by H. Unfortunately, this information about proportion of correct categorizations cannot be obtained in practice, except for uninteresting games like tic-tac-toe or oversimplified problems such as symbol recognition using small visual grids. At this point induction seems a doubly impossible problem: Hypotheses are much too numerous to consider even a fraction of them, and they cannot be evaluated anyway.

If objective probability measures are unobtainable, what alternative do we have for defining μ? One step toward an answer is to consider μ(H) as a *subjective probability*. Then μ(H) is the *credibility* of hypothesis H, i.e. our degree of *belief* in H (Cheeseman, 1985, Watanabe, 1969). With this broader interpretation of probability as belief, μ may be obtained from a combination of several sources: partly from feedback about task performance using H, and also from biases favoring certain kinds of hypotheses. The various kinds of knowledge which influence belief in a hypothesis have been called *knowledge sources* (Erman, Hayes-Roth, Lesser & Reddy, 1980).

There are two kinds of credibility: *evidential* and *extra-evidential*. Watanabe (1969) also called these *confirmation* and *creditation*. We shall sometimes use the term "confirmation" as a synonym for evidential credibility, although this can be misleading unless we realize that evidential criteria may guide hypothesis formation as well as confirm the final product.

There are many examples of both kinds of credibility. One definition of confirmation is the objective probability we discussed earlier, perhaps accelerated by sampling. As for the extra-evidential component, it might appear as a preference for implicit disjunction (see Figure 2). We shall present other possibilities later.

Evidential and extra-evidential criteria may be combined formally, to give a single measure of credibility (Watanabe, 1969, chap. 4). While Mitchell (1980) has called the implementation of either kind of credibility a bias, we usually consider extra-evidential credibility to be a special, goal-directed predisposition. Notice that evidential criteria correspond loosely to the AI notion of data-drivenness, and extra-evidential criteria correspond to model-drivenness. Methods for defining and

[3] The matter of hypotheses assessment raises some subtle issues. First, note that assessment takes place at two levels: (1) H evaluates (identifies) objects of the universe in the problem domain, and (2) μ evaluates (assesses) H, which is one of a set of hypotheses. Secondly, a graded assessment function μ is more useful than a rigid binary one in many domains. For example, Newtonian physics is much more useful than Aristotelean physics, but not always as good as relativity. Finally, we might ask how to evaluate an hypothesis which is expressed as a prototype. This is addressed in Part 2 of the paper.

updating credibility are given in (Lenat, 1983; Michalski, 1983; Rendell, 1985a; Wise & Henrion, 1985).

As we shall see in the next section, credibility may serve not only to assess hypotheses, but also to select them, and hence to speed induction. Before proceeding, let us summarize the main ideas of this section.

Definition 6. Hypothesis assessment and credibility

Because of the ambiguous nature of induction, many hypotheses compete, and we wish to compare them. Ideally, the *assessment function* μ would objectively measure the proportion of objects correctly identified by hypothesis H. Since exhaustive measurement is usually infeasible, we replace or combine objective probability with the subjective probability or belief, to form a composite assessment called the *credibility* μ(H). Because of uncertainties in the environment and in our interpretation, μ has many factors, such as goal-related feedback from the environment and predispositions or biases about which kinds of concepts may be good.

Definition 7. Inductive efficacy

An inductive method is *effective* if it produces hypotheses having high credibility, and hence concepts which are useful. We would like to assess credibility objectively, and we can afford to test hypotheses if the method does not generate too many. However, if the method is to be truly effective, it must apply subjective credibility to restrict hypothesis formation, and these subjective beliefs must correlate well with objective measures.

1.3 Schemes for efficiency: guidance from bias and goal-direction

An efficient induction algorithm should not only produce hypotheses which are credible, it should discover them in a reasonable length of time. In this section we shall examine some efficient methods for hypothesis formation. There are two basic ways to improve the efficiency of induction, one is model-driven, a product of bias or extra-evidential credibility, and the other is data-driven, a product of goal-orientation or evidential credibility.

1.3.1 Four kinds of bias

There are many instances of bias. To comprehend its manifestation in various structures and algorithms, we take an abstract perspective and relate all the approaches directly or indirectly to the hypotheses assessment function μ (Table 1). In terms of μ, there are two main varieties of inductive constraint and two kinds of each. One major bias is the exclusion of some hypotheses; this corresponds to limiting the domain of μ or deleting rows in Table 1(c). This restriction can be imposed from the outset or it can be governed dynamically by an inductive algorithm. Instead of precluding hypotheses, a more flexible scheme involves preferential ordering and informed selection of hypotheses; μ then becomes an evaluation function for search. Like exclusion, this can be either static or dynamic (the latter involves continual redefinition of μ). Each of these four kinds of bias can be implemented in several ways.

Universe and language restriction (narrowing the domain of μ). The easiest way to simplify induction is to begin with a small universe. This has to do with the degree of abstraction or grain size of objects. For example, the range of an attribute can be arbitrarily compressed, as in Samuel (1967). In Example 2 (symbol recognition), increasing the granularity of a visual grid would sacrifice resolution; alternatively a more severe kind of abstraction is to replace pixels (primitives) with strokes and curves (features). A similarly drastic choice is possible in Example 3 (checkers), where there are at least two major types of attributes: primitives giving the contents of each square of the board, versus features like piece advantage, center control, etc. As we saw earlier, the choice of abstract features reduces the number N of expressible objects, and consequently the number m^N of possible hypotheses when learning m classes (though the features may not discriminate adequately, resulting in unresolvable classification errors).

Another easy way to decrease the number of hypotheses is to limit the representation language for concepts. Without constraint, the full predicate logic permits relationships among objects. In this case, hypotheses describe a more complex universe than the original universe of objects. However, if logic is restricted to attributes only (i.e. to one place predicates), relationships are precluded, and the number of expressible concepts is diminished. Of course, this very restriction permits fast induction and is valid in cases where object interrelationships are irrelevant to the current problem. Restricting further, we may limit hypotheses, e.g. to a few conjunctions of attribute ranges $(a_1 \leq x_1 \leq b_1)$ & $(a_2 \leq x_2 \leq b_2)$ & .... This sort of syntactic constraint was illustrated in Figure 2 and discussed as a form of implicit disjunction. We shall detail these ideas later; also see Banerji (1985) and Michalski (1983).

Sets of hypotheses organized into spaces (actively zeroing some μ values as more is learned. Since induction is class formation, it can theoretically be attained by simple enumeration and testing. This is impossible in practice, but there are schemes to speed search for credible hypotheses. One general approach is to impose some structure on concepts and hypotheses which captures the nature of the problem, using knowledge about logic as in Figure 1, or about the domain as in Figure 2. The imposed structure can speed an inductive algorithm because such an algorithm does not search an amorphous set of classes, but rather a space of interrelated concept descriptions.

To see one way to implement this, we need to examine the logical structure of concepts (Figure 1). Suppose there are N objects in the universe and just two classes (Example 1, a case of "single concept" learning). If we name the objects $o_1, o_2, \ldots, o_N$, then any one of the classes H may be described as a disjunction of some of the o_i's. If we discover that the correct concept C is inconsistent with some hypothesis H, then every description H_i above H is also inconsistent with C (and the credibilities $\mu(H_i)$ become zero). For example, if $C = o_1 \vee o_2$, and if we reject o_3, then $\mu(o_3) = \mu(o_2 \vee o_3) = 0$. Mitchell (1978) has called this method candidate elimination; it is useful if N is small enough that the lower and upper boundaries of the version space are manageable.

In a version space, we interrelate concepts of the universe and we transform them using generalization and specialization operators such as those shown in Figure 1. In this diagram, moving up the lattice adds an object and moving down removes one. Instead of the simple lattice of Figure 1, a different structure may be imposed to interrelate concepts, i.e. alternative spaces may be defined. Rather than adding or subtracting a single in one step (Figure 2), we may use generalization and specialization operators which add or subtract a number of objects in a single application. Various forms of this appear in the machine learning literature (Duda & Hart, 1973; Langley, 1985; Michalski, 1983; Rendell, 1981).

One such possibility is shown in Figure 3, where an hypothesis represented as a conjunction of attribute ranges (i.e. a feature space rectangle) may grow (generalization) or shrink (specialization).[4] These operators may admit inconsistency and incompleteness. The amount of search required for a credible hypothesis depends on the *intensity* of the operators, which is the degree to which objects are added to or removed from the hypothesis in a single application.

As generalization or specialization operators are applied to some concept, a selection of new hypotheses $H_1, H_2, \ldots, H_q$ is produced, and some of them $H_{i-1}, \ldots, H_q$ may immediately be dismissed because they are judged to have little credibility compared with the competing hypotheses $H_1\ H_2, \ldots, H_i$ which are their

[4] Note that from the viewpoint of the overall process of concept formation from data, even specialization contributes to induction. This is in keeping with ideas that generalization and differentiation are two aspects of the same phenomenon (Koestler, 1964).

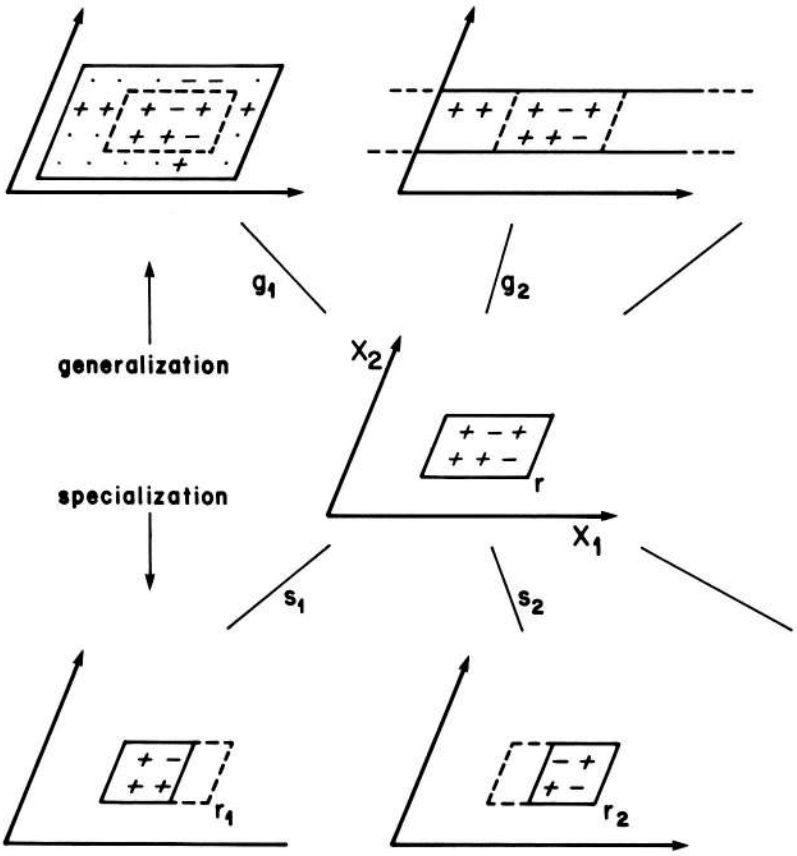

Figure 3. A more constrained version space (this is essentially another view of Figure 2). Here two factors speed induction. One is the syntactic limitation wherein concepts may only be conjunctions of feature ranges (rectangles). The other aspect is the greater intensity; these generalization and specialization operators may add or subtract more than one object at a time (see text).

siblings (this is beam search). Essentially the credibilities $\mu(H_{i+1}), \ldots, \mu(H_q)$ of these poorer hypotheses are zeroed.

Notice that the simplest propositional lattice (Figure 1) assures correct inferences. In other words, the associated credibilities are certain and not merely subjective, as long as the data are correct. On the other hand, there is no provision for noise or other forms of uncertainty discussed in Section 1.2. In contrast, when representation is restricted (Figure 2) or when the specialization or generalization operators are more intense, the situation becomes probabilistic (both of these biases occur in Figure 3). Such situations may cause error but they also may survive noise. To accommodate uncertainty we need subjective probability, belief, or multivalued credibility.

Hypotheses ordered by credibility (μ as evaluation function). We may exploit the credibility μ as an evaluation function to permit best first search in a space of hypotheses. An example of such a μ is some measure of simplicity. In this case the generalization labeled g_2 in Figure 3 might be awarded a high value since it does not require the specification of attribute x_1. Here credibility is extra-evidential.

Instead, μ may incorporate evidential credibility. The data may agree better with one hypothesis H_1 than another H_2; for example in Figure 3, the specialization labeled s_1 is preferable to s_2 because there are proportionately more positive examples in $H_1 = r_1$ than in $H_2 = r_2$. Based on current data, the probability of a positive example in r_1 is $p_1 = 3/4$, while $p_2 = 2/4$. These two probabilities can be considered the credibilities μ_1 and μ_2 of the hypotheses r_1 and r_2, or p_1 and p_2 can modify the credibilty of the original hypothesis r. "Best fit" criteria much like this are used by a number of systems, including CLUSTER/2 (Michalski, 1983), ID3 (Quinlan, 1983), and PLS1 (Rendell, 1977). This is goal-direction (Rendell, 1983a). CLUSTER/2 combines evidential with extra-evidential criteria to form a single "lexicographical" evaluation function (LEF).

Another instance of evidential credibility μ as evaluation function appears in Holland's (1975) genetic algorithm considered as an inductive system. Here hypotheses in a population compete for survival and are awarded various credibilities (fitness) μ based on their relative performance in a task environment. For example if three hypotheses H_1, H_2 and H_3 are observed to have relative performances of 6, 4 and 5 units, their credibilities might be normalized as $\mu_i = \frac{1}{2}$ (performance of H_i) / (average performance), giving $\mu_1 = 0.6$, $\mu_2 = 0.4$ and $\mu_3 = 0.5$.

Hypotheses (re-)ordered by continually changing credibility (μ as redefinable merit or agenda function). A more interesting and powerful kind of bias is provided by dynamic credibility. For instance, in a genetic system the more credible hypotheses continually replace the poorer ones, and this competition results in increasing quality over repeated generations. However, as shown in the example above, μ is computed for current data alone, so the assessment is always relative. In other words, the computed credibility depends on the state of our knowledge. Over several generations, our assessment becomes more demanding, as we insist on evaluating based on the best we know at the time. Credibility is thus a conditional probability Pr(hypothesis is correct | current knowledge). Conditional probabilities may be updated using standard statistics (Cheeseman, 1985), or related methods (Wise & Henrion, 1985). Another example of evolving μ is Lenat's (1983) Eurisko, in which the credibility may be altered both by the program and by the user.

Summary. We have related bias to credibility μ, an evaluation function over the set of hypotheses. These various biases may be encoded in the data structure, in the representation language, or in the inductive algorithm. Our choice of bias depends on knowledge about the task domain, and we can supply a large amount of it from

the outset. However, if too much bias is static, the program has little scope, and most of the power is not in the program. Ideally "bias control" would be a capability of a "higher-level" algorithm which would modify μ dynamically.

1.3.2 Efficiency through goal-direction

Utility as function, and induction as discovery of the function. Instead of viewing inductive efficiency as obtainable through various restrictions, biases or models, we can take a complementary perspective. Induction always has a purpose or goal, and the objects of a universe contribute to varying degrees. The degree of goal-satisfaction of an object **x** is its *utility* u (Rendell, 1983b).[5] One interpretation is that u is the credibility of the hypothesis "object **x** is in the desired class", or "**x** will contribute to the goal". We could just as well say that u gives the degree of class membership of **x**, or that u is the probability of **x** is a positive example of a concept. There is often no need to distinguish between these interpretations.

In supervised learning, an object **x** is presented as a positive or negative example of a class; thus the utility u(**x**) is 1 or 0. If this information is unreliable, u(**x**) is some (probability) value intermediate between 0 and 1 (Rendell, 1983a). Given a number of training examples **x**, we wish *to induce a function* u(**x**) for any object $\mathbf{x} = (x_1, x_2, \ldots, x_k)$. If the concept learning is rigid, u is a Boolean function, i.e. a logical expression as in Figure 4(a). If the learning is flexible, u is real-valued, e.g., a probabilistic expression as in Figure 4(b) or (c). We may select the type of function to be learned independent of data reliability.

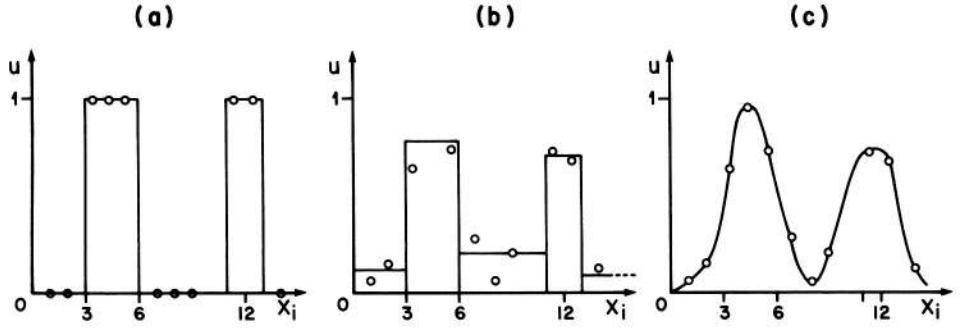

Figure 4. Utility of domain objects **x** or degree of class membership of **x**, as a function u(**x**). These three diagrams show just one attribute. In (a), u gives a binary decision about class membership, based on a logical model. In the other two cases, u gives graded belief, based on a probabilistic model. In (b), the model involves discrete probability classes; in (c), the model is a smooth curve. For any of these models to be effective, neighboring objects must have similar probabilities of membership (i.e. the implicit disjunction bias must be valid).

[5] Recall that objects can be anything, physical entities, operators, etc. Examples of the utility u include the "A-ness" of a visual grid, the strength of a game position, etc.

Efficiency, gradual discovery, and utility/bias synergy. Particularly in the probabilistic case, the data may be used to guide the induction itself. While the second major part of this paper explores this notion in depth, we may briefly state some of the ideas here.

Limited information about the variation of u with available training examples **x** permits tentative hypotheses: not necessarily about all the details of u, but about the form of this function (i.e. the model or bias), and perhaps about its rough outline (e.g., a step function). A simple example is shown in Figure 4, where there are two distinct areas of likely class membership. The limits of the concept $(3 \leq x_i \leq 6) \vee (11 \leq x_i \leq 13)$ in Figure 4(a) could be learned efficiently using the implicit disjunction bias with two neighborhoods (Figure 2). But this bias, itself, may be indicated only after probability classes are clustered as in Figure 4(b), and the system discovers that two neighborhoods are sufficient. In other words, the data may contribute to efficient bias selection. Further, goal-orientation and bias may combine to support and strengthen hypotheses both about the bias selection and also about the domain.

The observation and use of object utility obviates detailed generation and testing of hypotheses. In our earlier discussion of the illustrative examples, an hypothesis H was an entire utility function u, complete in every detail. In other words, H specifies u for every object x. In contrast, the incremental construction of u means that H is *built as it is tested*, and hypotheses of low credibility never appear.

Two kinds of hypotheses and the importance of probability. Notice that we have hypotheses on two levels, and beliefs about each. The utility $u(\mathbf{x})$ is the degree of belief that **x** is useful for the goal, and the credibility μ is the belief that the *function* $u(\mathbf{x})$ is correct. To avoid confusion, we shall continue to refer to the belief in the hypothesis H "function $u(\mathbf{x})$ is correct" as its credibility $\mu(H)$; and we shall refer to the belief in the hypothesis "object **x** is useful for the domain goal" as its utility $u(\mathbf{x})$.

Note that *both* kinds of hypotheses are functions. The utility function u may be binary-valued, giving a definite decision about class membership, or u may be real-valued, representing probability or degree of class membership. The non-binary choice converts to a binary decision at any time, yet the gradation facilitates incremental learning (Figure 4). Similarly, we recall that an hypothesis may be assessed in a rigid fashion, or it too may be flexibly represented and updated as a belief or credibility (Table 1).

To summarize this preliminary discussion of utility and bias, we note that there subtle relationships among bias, utility, efficiency, and incremental learning. We also suggest that humans first learn probabilistic information (Figure 4(b)), then convert it to a logic form for concise communication (Figure 4(a)), and further, that both representations are normally present at once. (See Glass & Holyoak, 1975; Medin & Smith, 1984; Rendell, 1983b.)

Definition 8. Utility

Utility is a term used to describe a goal-oriented quantity u related to the problem domain. For present purposes, we consider u to be the utility of some object **x**. The range of u may simply be binary, to indicate class membership, or u may be a real number in [0, 1], to code the degree of class membership. Given this flexible choice, we often do not need to distinguish between the degree of membership of **x** and probability that **x** satisfies the domain goal.

Since **x** is a vector of attributtes $(x_1, x_2, \ldots, x_k)$ u is a function of these k variables. *The problem of induction is to discover this function.*

The utility u(**x**) is our degree of belief in the hypotheses "object **x** is in the desired class". Here u is supposed be be correct, so efficacy is implicit. Compare this with the higher level hypothesis "function u is the correct one", where efficacy is explicit, and another function μ(u) measures the efficacy of u (see Section 1.2).

1.4 Understanding the contributions of various sources of knowledge

Given a number of diverse ways to improve the power of inductive systems, we might ask, as have Ritchie and Hanna (1984), whether we can understand these systems better. We would hope, for example, to be able to compare the amount of knowledge added by a program versus the amount given by the user. But is this possible? Here we take some steps toward an answer to this and related questions. We define a measure for the amount of generalization Γ, and then use Γ to determine the difficulty of an inductive problem, the contribution of a knowledge source, the strength of an inductive bias, and the amount of learning done by a system (cf. Michie, 1977).

1.4.1 Amount of generalization γ and inductive gain Γ

Consider the notion of generality/specificity, assuming that just one concept is to be learned. Given a universe U of objects, one concept C describing class $S_C \subseteq U$ is said to be more general than another B describing class $S_B \subseteq U$ (and B is more specific than C) if C applies in more cases than B, and if $S_B \subset S_C$; cf. (Dietterich, 1982, p. 365). For example, in Figure 1, the concept $C = (o_2 \vee o_4)$ applies in two cases and $B = o_2$ in only one, so C is more general than B. In Example 3, the number of winning states in checkers is greater than the number of board configurations having a piece advantage of 10 men (since there are more ways to win than by having an overwhelming piece advantage, and the latter should always win). Hence the concept "winning position" is more general than the class "piece advantage = 10".

In performing induction, a learning program must infer a concept C from a

universe of more specific descriptions $B_1, B_2, \ldots, B_N$. These may or may not be the fundamental objects of the universe. For the checkers example, C might be "winning position' . The B_i might describe all $N = 10^{18}$ individual states, or at another extreme, the B_i might be $N \simeq 20$ degrees of piece advantage (recall that an object can be anything).[6] In general, the smaller the value of N, the less difficult is the induction required to produce C. If the B_i are already abstract, if they have a large grain size, the induction is easier.

Aside from the number N of initial classes, there is a second source of difficulty. If the language used for the descriptions $B_1, B_2, \ldots, B_N$ is much different from the language used for C, particularly if the initial descriptors or *ground attributes* are different, then the induction is harder. For example, ground attributes for games such as checkers are often features like piece advantage, center control, etc. (the number of feature descriptions is then about 10^6 (Samuel, 1967)). Induction is feasible using features, mainly because they relate directly to winning; cf. the "single representation trick" (Dietterich, 1982). On the other hand, if we describe a state in terms of primitives that are the contents of each square of the board, the problem is qualitatively more difficult (Rendell, 1985b). Hence, our measure of inductive difficulty should account for the structuring required to create C from $B_1, B_2, \ldots, B_N$. For the present, however, we consider a first approximation of a proper measure.

We define the *inductive compression* γ between $B_1, B_2, \ldots, B_N$ and C (or between the classes they describe) as $\log_2 N - 1$ (the subtraction relates to the fact that the goal is two classes and $\log_2 2 = 1$). In the checkers example, if 10^6 feature descriptions are classified into "wins" and "not wins", the inductive compression $\gamma = \log_2 10^6 - 1 = 19$ bits. One reason for using the logarithm is that the amount of generalization is often extreme. Another reason is that inductive compression should be additive in cases involving multiple stages (Watanabe, 1969).

It may be that more than two classes are to be formed at a time. For example, rather than classifying game states as wins or not, we may wish to categorize states into classes of different strengths or utilities. With $m = 100$ utility classes, the overall inductive compression is $\log_2 (N/m) = \log_2 10^6 - \log_2 100 = 13$ bits. This suggests that it may be easier to spread the states out among 100 utility classes than to decide precisely which are wins.[7] On the other hand, learning unrelated classes is more difficult, not less, so γ is suitable only for learning degrees of utility, i.e. for learning

[6] Calling a particular piece advantage an object is really not so strange. Consider that we filter out information unnecessary for the problem at hand, such as whether a man is placed neatly in the middle of the square it occupies. Of course, classifying on the basis of piece advantage alone would result in many errors. But if piece advantage really were sufficiently descriptive for winning, we could legitimately filter out other details about the board configuration.

[7] Consider that an uncertain case might be resolved by placing it in a new class midway between two contenders. Also consider that γ is designed primarily for ground attributes which are tame features not requiring structuring, so that the bias of Figure 2 may be used to discriminate utility into intermediate probability classes that are adjacent in feature space.

the degree of membership for a single class. The suitability of γ of course extends to utility classes which may be only partially formed, as long as the class formation moves toward the ultimate goal.

Definition 9. Ground attributes

A *ground attribute* is an attribute explicitly defined and supplied by the user of an inductive system. Sometimes the program itself may create more abstract attributes (new terms) from the original ground attributes, though few systems have this capability.

Definition 10. Inductive compression γ (first approximation)

If there are N object descriptions $B_1, B_2, \ldots, B_N$ in the universe, to be formed into m utility classes $C_1, C_2, \ldots, C_m$, the *inductive compression* is $\gamma = \log_2 N - \log_2 m$.

Note that γ depends on the abstraction of the initial descriptors or ground attributes (reflected in N), and also on the spread of the end product (the number m of utility classes).

This definition, which assumes equal prior probabilities of the N objects, is the *information compression* (Watanabe, 1972). A better measure of γ should account for the fact that some of the objects may have little or no chance of occurring; this would preclude false impressions of the real problem difficulty. Other extensions of the definition would account for effects of feature interaction, and for the difficulty of rearranging, conceptualizing, or structuring objects, i.e. of transforming their descriptions if necessary. However, inductive compressions are often so large that estimates are indisputable. Furthermore, when we compare domains or systems, qualitative differences in inductive compressions can be resolved in favor of the lesser compression, so that inequalities remain valid.

The measure γ implicitly assumes that the a domain object has been placed in its proper utility class, i.e. that the inductive compression is effective. If we want to make efficacy explicit, we may instead consider induction as a search for the *correct means* to classify domain objects. In this case, we need to place each hypothesis into its proper credibility class. In Figure 1, the number of objects $N = 4$, and if we are learning one concept, $m = 2$, so the inductive compression $\gamma = \log_2 4 - \log_2 2 = 1$ bit. The number of hypotheses is $2^N = 2^4$, and if we are learning a binary credibility, the number of classes $M = 2$, so the *inductive gain* $\Gamma = \log_2 2^N - \log_2 2^N - \log_2 M = N - \log_2 M = \log_2 16 - \log_2 2 = 3$ bits. Somewhat like the situation with γ, a more complete measure of Γ should account for the difficulty of structuring the hypotheses

themselves. Unlike γ, Γ does account for the difficulty of finding an effective hypotheses (the *correct* concept). This is another way of saying that Γ accounts for any structuring required to classify the original objects of the domain universe (consider that an hypothesis is a utility function).

Definition 11. Inductive gain Γ *(first approximation)*

If there are 2^N hypotheses $H_1, H_2, \ldots, H_{2^N}$ to be formed into M credibility classes $\mu_1, \mu_2, \ldots, \mu_M$, the *inductive gain* is $\Gamma = \log_2 2^N - \log_2 M = N - \log_2 M$.

While we have written the number of hypothesis 2^N in terms of the number of domain objects N, the value of N is not always known. For example, when we use Γ for bias determination, we know the number of hypotheses, but not necessarily the effect on the objects of the problem domain (examples appear later). The M credibility classes may be only partially formed, as long as they are consistent, i.e. as long as any bias used in their formation is appropriate.

This single measure Γ can be used for the difficulty of an inductive problem, for the amount of learning done by a system, and for the strength of any sort of bias. The idea is that knowledge can be supplied by various means and agents, but the overall inductive gain is independent of the source.

1.4.2 Broad use of Γ *to assess difficulty, bias, and knowledge acquired*

As we have seen, there are many ways to constrain induction. These restrictions or biases involve several choices, including ground attributes, description language, hypothesis space operators, credibility criteria, and means for updating credibility. Each choice restricts and focuses, and the contribution of each may be measured by Γ. Let us consider some knowledge sources.

Problem domain and object description. To begin with the most obvious effect, the difficulty of a generalization problem depends on the grain size of the ground attributes relative to the number of classes to be learned. Since the inductive gain is $\Gamma = N - \log_2 M$, where m is related to the purpose of the induction and N is number of objects in the universe, increasing the initial grain size decreases the value of Γ. As a simple case of Example 1, suppose there are two attributes and each may take 8 values. The number of objects in the universe is then 64, and the number of hypotheses is 2^{64}. The inductive gain required to learn one concept is $\Gamma = \log_2 2^{64} - 1 = 63$ bits.

Syntactic constraint. Next, the representation language may restrict hypotheses and thereby reduce the effort necessary to produce and test them. Continuing with our example, suppose that the only concept descriptions permitted are conjunctions of the form $(a_1 \leq x_1 \leq b_1)$ & $(a_2 \leq x_2 \leq b_2)$; these descriptions are the rectangles of Figures 2 & 3 producing the bias of implicit disjunction. The number of sensible descriptions of this type is $(\Sigma 8)^2 = 1296$. $\text{Log}_2 1296 = 10$ bits, so the gain Γ provided by our syntactic constraint is $64 - 10 = 54$ bits, leaving only $63 - 54 = 9$ to be learned. (Assuming that such language restriction is appropriate, nearly all the gain is provided by the knowledge that a single conjunction is adequate).

System learning versus user hints. Unless the user provides advice, the remainder of the induction is performed by the system, and we may compute the system contribution by subtraction. There are some remaining considerations, however. The user may alter the credibility μ or its equivalent. This user aid may sometimes be measured from the number of candidate hypotheses eliminated, though not always. If μ is dynamically altered by the programmer (Lenat, 1983), it is difficult to apportion credit between user and machine (Ritchie and Hanna, 1984). Despite this unresolved difficulty, there are many learning systems for which credibility is determined solely by the program, and for these learning systems, we have:

Definition 12. Problem difficulty, bias strength, and conceptual knowledge acquisition

The total inductive gain $\Gamma_d = \Gamma_o + \Gamma_s$, where Γ_o is the initial gain provided by the restriction of universe and language, and Γ_s is the gain of the learning system. The total gain Γ_d is the *problem difficulty*. Γ_o is a measure of the *bias strength* which was discussed qualitatively by Mitchell (1980), and by Utgoff and Mitchell (1982). Γ_s is a measure of the amount of induction done by a system, the *conceptual knowledge* acquired (Rendell, 1983b).

Conceptual knowledge Γ_s can be acquired in stages (Rendell, 1983b). Γ_s can be used in combination with other measures to determine the power of a learning system. (Consider an analogy with physics, where power = energy / time. Roughly speaking, if Γ_s measures energy expenditure, then the inductive power = Γ_s / computer time.)

2. Realization of inductive power using strong initial bias

This major part of the paper examines some schemes and principles for attaining powerful induction, using a particular bias sometimes called *selective induction* (Michalski, 1983). Loosely speaking, selective induction means implicit disjunction, and this requires "tame" ground attributes giving well behaved utility functions (see Figures 2 & 4). Selective induction is a prevalent, useful and extreme bias; it is widely used because it has significant scope and markedly improves efficiency. To investigate the characteristics of selective induction, we shall use the fundamentals discussed in Part 1. We shall also examine some new issues, such as judicious management of uncertainty to facilitate incremental learning and resource conservation. There are three sections. The first explains what selective induction is, how it simplifies the problem in terms of inductive gain Γ_o, and when selective induction is appropriate. Section 2.2 develops an approach for managing uncertainty through the use of probability. The final section considers the issues of efficiency and efficacy and examines some specific methods and principles.

2.1 The syntax and semantics of selective induction

As Utgoff and Mitchell (1982) have argued, there are two important aspects of any bias. One is the strength of the bias, which we have defined as Γ_o, and the other aspect is the appropriateness of the bias, which we have also called its efficacy. In order to know the bias strength imposed by a language for concept representation, we need to examine the *syntax* of that language. In order to understand the appropriateness of a bias, we must examine the meaning or *semantics* of the constraint imposed by the language.

Preliminary definition. Selective induction

Selective induction is a commonly used bias which has many guises. Its essence, however, is that the user chooses ground attributes so that implicit disjunction will be sufficient (see Figure 2). Either literally or metaphorically, the problem of induction is reduced to *selecting* neighborhoods of feature space. Implicit disjunction permits relatively straightforward, efficient algorithms which restrict the number of elements required to specify a concept (these elements may be disjuncts, prototypes, or parameters of various sorts). These notions will become clearer as we proceed.

2.1.1 The syntactic limitation and bias strength of selective induction

We may consider selective induction as a restriction of predicate logic. The full predicate calculus permits the expression of structure or relations through predicates having more than one argument, but if only unary predicates are allowed, relationships cannot be expressed among objects in the universe. What remains is the capability to identify attributes of an object (see Definition 1); this helps to construct classes of objects. We express an object as a vector $\mathbf{x} = (x_1\ x_2, \ldots, x_k)$, and we describe a class as a concept C based on such vectors, e.g. $C = (3 \leq x_1 \leq 6)\ \&\ (1 \leq x_2 \leq 5)$ as in Figure 5(a).

A related issue is the type of scale used for attributes. Two possibilities are nominal and interval scales. A nominal scale assumes no order in the values of an attribute, whereas an interval scale does. In combination with an inductive algorithm, the ordering may restrict the number of describable hypotheses, so an interal scale can increase the bias strength Γ_o (i.e. the inductive gain provided by the user). We shall primarily discuss interval scales (this includes the binary possibility, which is a special case of both interval and nominal scales). For an examination of various scales and their effects, see Anderberg (1973, p. 27 ff.). For experiments with nominal and structured variables see Michalski (1983).

Many algorithms exploit interval features and proximity measures to restrict the number of hypotheses. These methods all construct feature space neighborhoods by aggregating vectors (generalization) or by differentiating sets of vectors (specialization). There are three basic ways to limit formation of classes (sets of vectors) by constraining concepts (class descriptions). One way is to circumscribe the positive class, another is to delineate the boundary between the positive and negative classes,

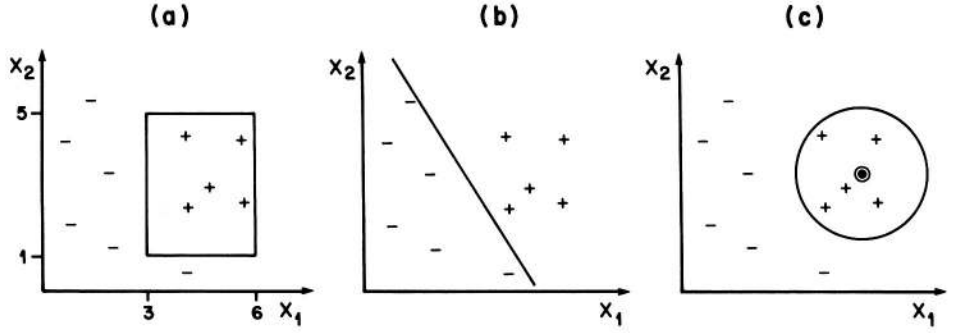

Figure 5. Three methods for selective induction, in which classes or concepts are concisely represented as neighborhoods of feature vectors. In (a), a (positive) class is described as a concept. In (b), the positive and negative classes are distinguished by the boundary between them. In (c), a concept is represented as a centroid or prototype, and class membership may be a decreasing function of distance from the prototype. Each method is a form of implicit disjunction (see Figure 2). When justified, selective induction provides great saving in computational resources because of the tremendous reduction in the number of hypotheses considered.

and the third method is to discover a "most representative" vector of the positive class, called a centroid or prototype (see Figure 5).

Methods which describe a concept. One kind of method, commonly used in AI, replaces individual attribute values with sets of values (Dietterich, 1982; Michalski, 1983). For example, if $\mathbf{x} = (x_1, x_2) = (4,0)$, one concept C which could be induced using $\mathbf{x}$ as a positive example is shown in Figure 5(a): $C = (3 \leq x_1 \leq 6) \,\&\, (1 \leq x_2 \leq 5)$. If the attributes have interval scales (as they do here), this selective expression is a rectangle in feature space (see also Figures 2–4). Rectangles are economical to store and to manipulate (Rendell, 1981).

Assume that a concept is to be learned as a function of two attributes. As in Section 1.4.2, suppose that the universe has 64 objects, covering a discrete 8×8 feature space. Let the description of a datum be (x_1, x_2), where each feature has eight values: $x_1 \in \{a_i\}$ and $x_2 \in \{b_j\}$ $(0 \leq i, j \leq 7)$. With no constraint, the number of ways of forming 2 classes is very large: $2^{64} \simeq 1.8 \times 10^{19}$, so that the required inductive gain is $\Gamma = \log_2 2^{64} - \log_2 2 = 63$ bits.

However, the allowable concepts may be restricted. One constraint is to impose the form $x_1 = a_{i_1} \vee a_{i_2} \vee \ldots \vee a_{i_n} \,\&\, x_2 = b_{j_1} \vee b_{j_2} \vee \ldots \vee b_{j_m}$ (where $n = m \leq 8$). Since the values for each attribute may be selected independently of the other attribute, the number of formulae of this type is $2^8 \times 2^8 = 2^{16}$, an extremely small fraction ($2^{-48} \simeq 10^{-14}$) of the total number when there is no constraint. Here the bias strength $\Gamma_o = \log_2 2^{64} - \log_2 2^{16} = 64 - 16 = 48$ bits. Michalski (1983) has employed this bias, which he calls *internal disjunction*.

More restrictive still is an earlier example, in which the only concept descriptions permitted are conjunctions of the form $(a_1 \leq x_1 \leq b_1) \,\&\, (a_2 \leq x_2 \leq b_2)$. As we saw in Section 1.4.2, the gain resulting from this bias is $\Gamma_o = 64 - 10 = 54$ bits. If we relax the constraint by permitting a disjunction composed of two such descriptions, we reduce the bias by about 10 bits (the logarithm of the number of hypotheses added), so this less severe bias is 44 bits.

These biases are all forms of implicit disjunction (Figure 2). Implicit disjunction has appeared in many learning systems which are superficially different. Samuel (1967) used rectangular feature space cells for checkers. Rendell (1985a) has constructed a complex scheme more flexible than Samuel's. Michalski (1983) has used interval, nominal and "structured" scales, all with a strong bias. Hunt, Marin & Stone (1966) and Quinlan (1983) have incorporated decision trees, a slightly different representation with the same underlying bias (as we shall see later).

Methods which express discriminating boundaries. The second way of representing a class is to specify algebraic relationships among attributes of objects. In its simplest form, this means interval features and hyperplane insertion. We can say that an object $\mathbf{x}$ is a member of the desired class if $\mathbf{x} = (x_1, x_2, \ldots, x_k)$ falls on one side of a hyperplane in feature space. In Figure 5(b), $1.6x_1 + x_2 > 8$ is the separating

boundary. This kind of representation is used in discriminant analysis (Duda & Hart, 1973; Tou and Gonzalez, 1974).

In linear discriminant analysis, the number of hypothesis is a function of the size N of the universe and the feature space dimensionality k (Duda and Hart, 1973, pp. 69 ff.). The bias becomes stronger as N increases.[8] In our example of 64 objects in an 8 × 8 space, the number of dichtotomies which can be produced by inserting linear boundaries is only about 4000, which is approximately 2.2×10^{-16} of the total number of possible categorizations. Here the inductive gain is about $64 - \log_2 4000 = 52$ bits (leaving only 11 bits to learn).

Methods which represent prototypes. The third common expression of a concept is its centroid or prototype, which is interpreted as the most representative object of the class. Its spatial extent is given by class boundaries which may be sharp or fuzzy. Sharp boundaries correspond to a single implicit disjunction, whose bias we have already quantified for rectangles (the center or centroid of a rectangle is its prototype). The bias is roughly similar if the class is spherical, if the boundaries are fuzzy, or if the attributes are nominal.

Both the prototype and the boundary representation admit analytic methods, which do not explicitly generate hypotheses, but rather compute the parameters of a model. Hence these methods can be even faster than our bias calculations suggest (although incremental learning is a problem). See (Duda & Hart, 1973; Medin & Smith, 1974; Tou & Gonzales, 1974). Easterlin & Langley (1985) discuss some advantages of prototypes. Rendell's (1983a, 1985a) systems accommodate incremental learning and yet they represent concepts as prototypes (and simultaneously as logic descriptions). Later we shall consider details.

Synthesis. What are fundamental similarities and differences among these three methods for selective induction? First, the commonalities: When attributes have interval scales, restricting classes by using discrimination boundaries is similar to constraining concepts by using conjunctions of attribute ranges. Both these and the prototype methods rely on feature space proximity (Figure 5), i.e. implicit disjunction (Figure 2). When attributes are not interval, the basic idea is similar, except the concept descriptions are conjunctions of attribute sets rather than conjunctions of attribute ranges. All three methods produce concise class descriptions, and they all embody a strong bias.

[8] More precisely, suppose that the N objects in the universe are evenly spread out in k dimensions. Then the fraction of dichotomies (single concepts) covered by a linear discriminant function is

$$f(N, k) = \begin{cases} 1 & N \leq k + 1 \\ \dfrac{2}{2^N} \displaystyle\sum_{i=0}^{k} \binom{N-1}{i} & N > k + 1. \end{cases}$$

This bias is selective induction. It limits the representable concepts to implicit disjunction, and this is describable as long as the problem fits the restriction. In other words, when the domain permits, we may aid efficiency without impairing efficacy, by imposing appropriate syntactic constraints to reduce the number of hypotheses expressible. The strength of this bias is measurable; as our examples suggest, selective induction often results in an inductive gain of 80% or more (and this measure is logarithmic).

While the three major methods share implicit disjunction, they also exhibit important differences. Discrimination functions may be nonlinear, and these and other techniques improve flexibility (they reduce the bias somewhat). Moreover, there are fast algorithms for computing decision boundaries and prototypes (Tou & Gonzalez, 1974). On the other hand, concepts represented in logic are easily comprehensible to humans (Michalski, 1983), and some associated methods are well suited to fast incremental learning (Rendell, 1981, 1983a). Like logic, the prototype representation is also comprehensible, but at the same time, it is naturally suited to noisy situations (Matheus, 1985).

As we shall soon see, these three basic methods may be combined to produce a more powerful scheme. Briefly, discriminant analysis becomes clustering when degrees of belief or probabilities are grouped. If the discriminant boundaries are parallel to feature space axes, the clusters become conjunctive concepts or rectangles (Figure 5). Each rectangle r expresses not a concept C in the problem domain, but rather the probability p that r implies C. Each (p, r) pair has a centroid, and a set of centroids takes the place of a single prototype for C.

2.1.2 The semantics of selective induction: preconditions of the bias

When is selective induction appropriate? Consider our illustrative examples. The first one has often been the subject of AI experiments, in which a concept such as "ball" is found to be round in shape (not square or triangular), and smooth in texture (not rough). Such attributes usually have few values and relate directly to the concept to be learned (Dietterich, *et al.*, 1982). Similar situations arise in many tasks, such as disease diagnosis (Michalski & Chilausky, 1980).

Special attributes. However, the problem is not always so simple. To contrast two extremes, consider two kinds of ground attributes in symbol recognition. One is abstract features, such as various curves and lines which compose a symbol to be learned. In this case there is a direct relationship between ground attributes and concept (the symbol). A different choice of attributes is an array of low level primitives which are squares (pixels) of an image. Here the relationship between attributes and concept is distant and complex. As another example, consider checkers, where an object (a board configuration or state) can once again be described in two fundamentally different ways. One choice of ground attributes is

the set of 32 primitives indicating the contents of each permissible square. In contrast, a more abstract description of a state involves features such as piece advantage, mobility, etc. Abstract features have a rather direct relationship to the ultimate concept (winning): a small change in the value of any feature causes a small change in the board strength. In contrast, primitives are not directly related to the goal: a small change in one square can make a drastic difference.

This returns us to the notion of *utility* discussed in Section 1.3.2. In learning the concept of "ball", a positive training example might be the vector (large, round, smooth); it has a utility of 1, whereas (large, square, rough) has a utility of 0 as a ball. In symbol recognition, the utility is the degree of match between a pattern and the target symbol. In checkers, the utility is some measure of the strength for winning. No matter what the problem domain, there is always some purpose or goal for induction; this purpose may be expressed in terms of the utility. As we saw in our earlier discussion, the utility is a function u of some vector of ground attributes **x** that describe the object under consideration. Our recent examples show that when **x** is a vector of primitives, the function u(**x**) is very irregular, but when **x** is a vector of abstract features, the function u(**x**) is well behaved. Abstract features prearrange objects with regard to their utility, so that u(**x**) has few peaks (Figure 4).

Utility coherence as a requirement for selective induction. In Section 1.3.2 we saw that the problem of induction is the problem of discovering the utility function u(**x**). To detail the requirements for selective induction, let us consider some possible forms of u. Suppose first that utility is binary, and indicates class membership. Figure 6 shows two different patterns of concept complexity. The situation in Figure 6(a) is an instance of implicit disjunction like that of Figure 2. In cases like Figure 6(a), one

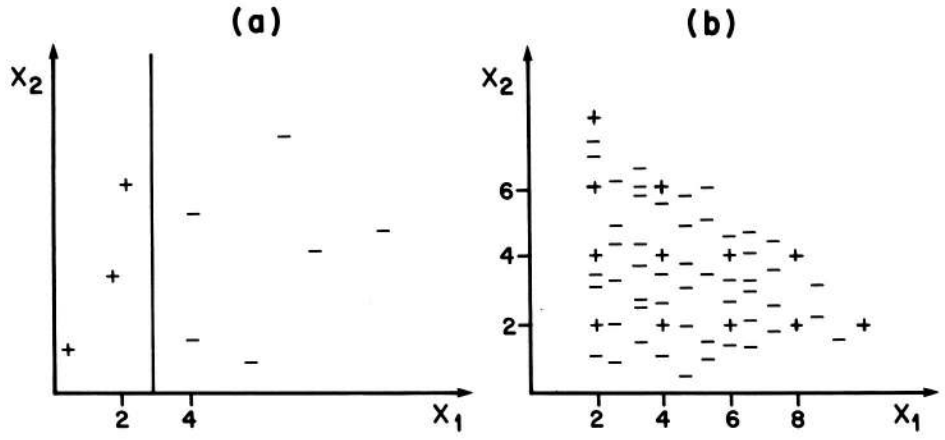

Figure 6. The special nature of attributes for selective induction. Selective induction (a) assumes a very straightforward kind of regularity, where neighborhoods are uniform because attributes are abstract features. In contrast (b), if objects are described in terms of less favorable attributes (primitives), any regularity is usually more complex, and selective induction is not appropriate if the pattern extends far.

of the fast methods for selective induction discussed earlier will suffice to produce a simple concept description.

For Figure 6(b), however, a little analysis proves that selective induction is inappropriate. According to Definition 7, the efficacy of selective induction would be low here because a small number of neighborhoods cannot express any credible hypothesis whatever. As we saw in Definition 6 and the discussion preceding it, the credibility of an hypothesis must ultimately depend on objective evidence expressed in terms of classification accuracy. In Figure 6(b), the evidence does not support a small number of neighborhoods, since any such hypothesis results in many classification errors.

Hence, to be appropriate, selective induction must work with attributes which permit the implicit disjunction bias of Figure 2. We can think of algorithms for selective induction as operating on individual windows of Figure 2 in an attempt to cover positive examples and omit negative ones. The windows become disjuncts of a concept, and if too many disjuncts are required, the process can be prohibitively expensive (Banerji, 1985).

Let us broaden our examination to include both binary and non-binary utility. In Figure 4(a), the utility u is binary; this diagram corresponds to Figure 6(a) except that only one attribute is shown and there are two neighborhoods of positive objects instead of one. Each blip in Figure 4(a) requires a separate disjunct in the corresponding concept description. In the other two diagrams of Figure 4, u is a real number representing the probability or degree of class membership. In Figure 4(b), the blips can be identified efficiently by fast algorithms for selective induction (e.g. discriminant analysis). In Figure 4(c), the maxima can also be located by fast methods (such as curve fitting). In contrast, if (b) and (c) had many peaks, the sparse data typically available would not support the models necessary (e.g. low order polynomials for curve fitting).

Whether the utility function $u(\mathbf{x})$ is binary, discrete, or continuous, selective induction requires uniformity or smoothness in $u(\mathbf{x})$, since neighborhoods of constant utility are discovered. Selective induction needs local utility coherence.

These ideas have arisen in other work. Koestler (1964) argued that goal-direction is necessary in (perceptual) induction. Medin & Wattenmaker (1985) have examined the importance of utility coherence in contexts like ours. Rendell (1985b) has extended the notion of utility coherence to achieve tractable induction in cases of concept formation where difficult structuring is required.

Summary. Since induction is purposeful, it aggregates patterns of utility. *Selective* induction discovers simple utility functions using special ground attributes which produce utility coherence. Utility coherence may appear as implicit disjunction or as smooth variation; the exact manifestation depends on the model imposed. Selective induction may complete logical models (constructing concepts of unit utility), it may find discrete functions (clustering classes of different utilities), or it may fit real

functions (parameterizing curves of continuous utility). Every kind of selective induction involves concise expression, strong bias, and efficient algorithms.

Definition 13. Selective induction

Syntactically, ***selective induction*** is class formation in which the classes are described as a small number of disjuncts (if utility is binary) or as a small number of maxima (if utility is continuous).[9] This means a strong (measurable) constraint or bias, and consequently fast algorithms. Semantically, if selective induction is to be effective (appropriate), a very special characteristic is necessary: the vector $\mathbf{x}$ of ground attributes describing objects in the domain must be abstract features that cause the utility function $u(\mathbf{x})$ to be uniform, smooth, or locally invariant.

2.2 Dealing with uncertainty: using exceptions to advantage

This section examines a general framework for selective induction. The result will be a probabilistic method which not only can manage uncertainty, but also can profit by it.

2.2.1 Binary versus probabilistic concept representation

Simple, all-or-none representation. In supervised learning (learning from examples), a teacher provides positive and negative training examples of a desired class. As discussed earlier, concepts describing the class may be formed in various ways: by inserting decision boundaries, by determining conjunctive descriptions, or by finding representative prototypes. Each of these forms of selective induction amounts to implicit disjunction, wherein a few disjuncts or neighborhoods compose the concept. In the simplest case, there is only one neighborhood; e.g. in Figure 7(a), an object $\mathbf{x}$ is an instance of C as long as $(0 \leq x_1 \leq 3)$. Discovery of a precise dichotomy into positive and negative aspects of a class is *binary induction.*

In cognition, this rigid, *defining features* view of concept representation has been contrasted with a more flexible perspective which accommodates uncertainty, called the *probabilistic* view (Medin & Smith, 1984). Humans may use both kinds of representations, the binary or defining feature one for fast communication (it requires less information), and also the probabilistic representation for noise management and simple modification (probabilistic information can be updated

[9] Michalski's (1983, p. 105) definition of selective induction is that the process creates no new attributes.

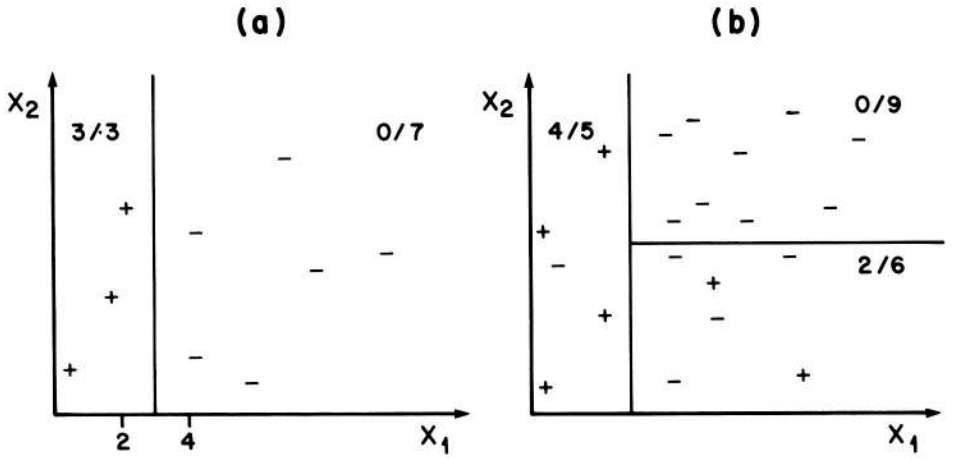

Figure 7. Binary versus probabilistic concept representation. Ideally, classes can be neatly differentiated into positive and negative instances (a). But more commonly, exceptions occur (b). In less extreme cases, where the neighborhoods still cohere, the exceptions are just anomalies whose effects can be recorded using proportions or probabilities.

reliably and easily). A probability can be converted to a binary decision at any time.

Flexible, probabilistic representation. But not only is the probabilistic representation more flexible, it is essential in a fully capable system. While simpler problem domains can avoid probability, the real world requires us to deal with uncertainty. As we saw in Section 1.2, sources of uncertainty include sparse and noisy data, imperfect ground attributes and syntactic constraints. Moreover, there are natural reasons for preferring some hypotheses over others, depending on bias and evidence. In addition, incremental learning situations adjust preferences: we must retain more than one hypothesis, and dynamically alter the credibility of each.

There are certain minimal requirements for updating credibility or belief. One requirement, especially useful in incremental learning, is not only a probability, but also a second order probability, or uncertainty in the credibility itself. In other words, we have both a belief and also a confidence in our belief. In Figure 7(b), a sample of $n = 5$ objects $\mathbf{x} = (x_1, x_2)$ have the description $(0 \leq x_1 \leq 3)$, and $g = 4$ of them are positive instances. This leads us to predict the probability of finding a positive instance in $(0 \leq x_1 \leq 3)$ as $u = g/n = 4/5 = 0.8$. But we might be more or less certain of any such judgment. It is possible to compute the confidence, or (inversely) the error e in u. With $n = 5$, e might be $\pm 50\%$, and this would reduce to about $\pm 13\%$ if n were increased to 20 (elementary statistics shows that e varies inversely with $\sqrt{n}$, although there are other factors).

In incremental learning we would discard the original data, but we can summarize the important information: probability and its error are the just pair (u, e). Our

interpretation of a high u and low e would be that there *must have been* few exceptions, and that our experience *must have been* extensive. If e were high, we would assume that our experience must have been limited, unless u were around ½, which would indicate many exceptions. If $u \simeq ½$ and e were very low, then we might suspect that the attributes are inadequate, since no binary statement can be made about the utility, and we are sure that further experience will not improve matters much. Rendell (1983a, 1985a) gives some methods for computing and updating beliefs for incremental induction.

Our discussion has assumed that just two numbers are adequate (the belief u and its error e). Other possibilities exist, however, since underlying the concise representation (u, e) is a complete probability distribution (though we rarely know what it is). Researchers have taken a number of related approaches to the problem, although a standard technique has not yet emerged (Cheeseman, 1985). This is reminiscent of work in knowledge representation, where alternative languages are sometimes equivalent, but one may be more natural for a particular application.

Below we consider a representation which captures both the usual requirements (e.g. the ability to identify a new object), and also the demands of uncertainty. These include the assessment of an hypothesis (its credibility or belief), the confidence we have in the assessment (or inversely, its error), and the ability to convert from probabilistic information to a binary decision while retaining the potential for updating (incremental learning).

2.2.2 A specific approach to uncertainty

Representing sub-concepts. Instead of just two categories (positive and negative), the presence of uncertainty requires gradation. Rather than expressing a goal concept C outright, we represent *probability classes*; i.e. classes of discrete values which represent different probabilities of membership or varying degrees of membership. Associated with a probability u may be one or more *sub-concepts*. In Figure 8, the leftmost rectangle is $r = (0 \leq x_1 \leq 4) \,\&\, (0 \leq x_2 \leq 2)$, and this sub-concept has probability $u = 0.2$ of being C. In other words, u is the conditional probability Pr(C|r) of C.

From one point of view, the number of classes is the number of values;[10] from another point of view, there are only two classes, positive and negative. In this binary view, the utility function is Boolean and directly determines class membership; in the probabilistic view, the utility function is multi-valued and indicates probability of class membership. In the binary view, we attempt to learn the ultimate concept C

[10] In practice, it is always reasonable to use a finite range for u. If the range is decided by machine (i.e. if there is no prior knowledge of probability classes), the learning is unsupervised in this respect, even though the training examples may be presented in a supervised fashion. One way of implementing the creation of probability classes (unsupervised learning) is by using clustering, which we shall discuss later.

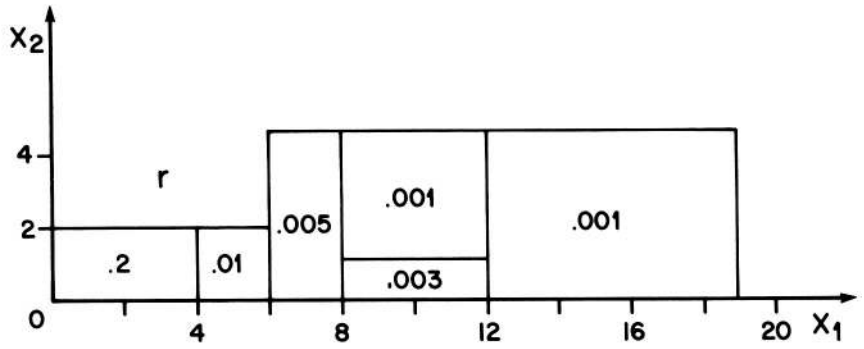

Figure 8. Probabilistic concept representation or region set. Instead of coding binary utility as a Boolean function, a concept comprises separate sub-concepts r of varying probabilistic utility u. (u values are shown inside rectangular sub-concepts r.) u is the degree of class membership or the belief in the hypothesis "an object falling within rectangle r fullfils the desired purpose". A tuple of associated information such as R = (r, u, . . .) is called a region. A feature space partition with this probabilistic information is a region set **R**. **R** is a concise, step-wise constant representation of a utility function u. Formed from regional sub-hypotheses, **R** embodies the complete hypothesis.

directly; in the probabilistic view, we learn intermediate, sub-concepts r for the eventual determination of C.

Inducing concepts and sub-concepts. To induce a binary concept, we would use one of the methods for selective induction discussed earlier, such as replacing constants with variables. As a simple example, suppose there is only one attribute, and restrict the form or model of C to the expression allowing just one interval: ($a \leq x_1 \leq b$). This is a simplification of Figure 4(a), where the utility u is a Boolean function, and u has the value 1 at just a few neighboring points. Given the model and positive and negative examples of C, a system can easily determine values for a and b. A slightly different view of C appears in Figure 7(a), which suggests the discrimination method. A program can easily find the parameters of the decision boundary.

Normally the discrimination ends after a single boundary has been found to delineate (most) positive and negative instances. But suppose we continue the discrimination when there are exceptions. Figure 7(b) shows this case; here we form several classes having different probabilities of class membership. The classes are sub-hypotheses, elements of a discrete utility function.

In cases of uncertainty, the logical view is replaced by a real-valued one; instead of a Boolean form, the model is a discrete or continuous function. If the model is discrete, sub-hypotheses may be discovered by aggregating the positive examples or by differentiating them from the negative ones, as in Figure 7. In this case, the eventual concept is a by-product of the search for local invariance, or of the search for marked dissimilarity. If the utility model is some curve, as in Figure 4(c), then the discovery of the parameters for the model, and perhaps the selection of the model

itself, may be a concequence of preliminary classification procedures in the discrete approach (Rendell, 1983a). These procedures are based on similarity and dissimilarity. We may conclude that the discovery of a concept is the result of a search for similarity or dissimilarity. In other words, hypothesis formation *follows* invariance localization or dissimilarity detection.

Housing sub-hypotheses in a region set to produce a concise but flexible representation of a concept. When selective induction is uncertain and incremental, a formative representation of a concept C may profitably be represented as multiple sub-hypotheses. For each sub-hypothesis, we store its probability u of membership in C, along with the error e in u. Sub-hypotheses are usefully expressed as rectangle r, but the prototype representation may also be worthwhile. In fact, both of these representations are important depending on context, so we retain both. To avoid ambiguity with the prototype of the ultimate concept C, we call a prototypical member of r its *centroid* c. Let us refer to the quadruple R = (r, u, e, **c**) as a *region*. As illustrated in Figure 8, a partition of feature space gives the *region set* **R** = {R}. A region set is a concise expression of the utility function, which is the hypothesis. An inductive system may perform various operations on regions, including differentiation (splitting reactangles r), generalization (merging rectangles r), reorganization (of **R**), and revision (altering the values of p and e); see (Rendell, 1985a).

Regions represent sub-hypotheses as rectangles and centroids, each for several reasons. First, there are a number of advantages of rectangles. Incremental learning demands progressively refined knowledge, and rectangles facilitate modification, since they can be split to form smaller rectangles or aggregated to form larger ones, as just suggested. Rectangles are easy to express, and convey meaning well. Further, the size of r in a region R = (r, u, e, c) indicates the rate of change of u near **c** (if many small rectangles appear in some neighborhood, the utility changes rapidly there). But centroids **c** may also be important, the precise reason depending on the particular application. For instance, **c** may be the best representation of a subconcept for class recognition, since similarity can be defined as a distance in feature space. Another use of the centroid is to determine the utility u as a continuous function of features; this vector **c**, along with the u value of a region R, define one datum in curve fitting: u may be an evaluation function as in Rendell (1983a).

There are other important aspects of a region set **R** = {R}. The complement 1 – u of the utility u gives the proportion of exceptions within a rectangle; this concise information may be used to decide when to add or create a feature (i.e. when to change the bias). **R** is intermediate between data and a binary decision, which allows continual revision of the utility function, yet permits a tentative binary decision at any time. A region R is economical and effective: the probability u is a summary of many training examples, each of which has some influence, but none of which may overwhelm the learning. The utility u and its error e are expressed as single numbers.

The next section develops these and related ideas about economy.

Definition 14. Regions and sub-hypotheses

The hypothesis *is* the utility function. A *region* R is a piece of information about the utility function: a local implicit disjunction or neighborhood with a uniform utility and error. A *region set* $\mathbf{R} = \{R\}$ is a discrete function composed of regions; it may represent both a binary function (the logical form of the desired concept), and also a continuous function (the very detailed form). Hence **R** is intermediate in generality. A region set codes the hypothesis; a region codes a sub-hypothesis. These structures aid time efficiency and cognitive economy.

2.3 Speed and cognitive economy: representations, methods and principles

Considerations of economy involve both storage and time, and arise both during performance (hypothesis use) and during learning (hypothesis formation). These aspects are related; for example fewer hypotheses and sub-hypotheses to store means fewer to consider during use, and also fewer to access during incremental learning. In this section, we examine various sources of economy, we compare their manifestations in learning systems for selective induction, and finally we propose some general principles for efficient and effective induction.

2.3.1 The optimal discrimination criterion

Regions or utility classes should not proliferate without good reason. The various rectangle sizes and shapes in Figure 8 suggest that a flexible strategy is required to accommodate different domain characteristics. Suppose we begin with one large rectangle; how do we know when and how to subdivide it? The answer is quite straightforward: we are interested in discriminating utility u in order to differentiate the quality of competing objects during task performance. Therefore we would like to discover maximally dissimilar utility classes.

The basic criterion. We might employ various algorithms, but to simplify, suppose we repeatedly split rectangles in two. In this case we measure and compare the probabilistic utility u for two tentative subrectangles. Figure 9 depicts one example for a set of $n = 287$ data, $g = 17$ of which are positive examples of a concept. Various insertions of a tentative boundary result in different values for the utilities $u_1 = g_1/n_1$ and $u_2 = g_2/n_2$ of the two subrectangles (where $g_1 + g_2 = g = 17$, and $n_1 + n_2 = n = 287$). The best dichotomy is the one that produces the greatest dissimilarity in

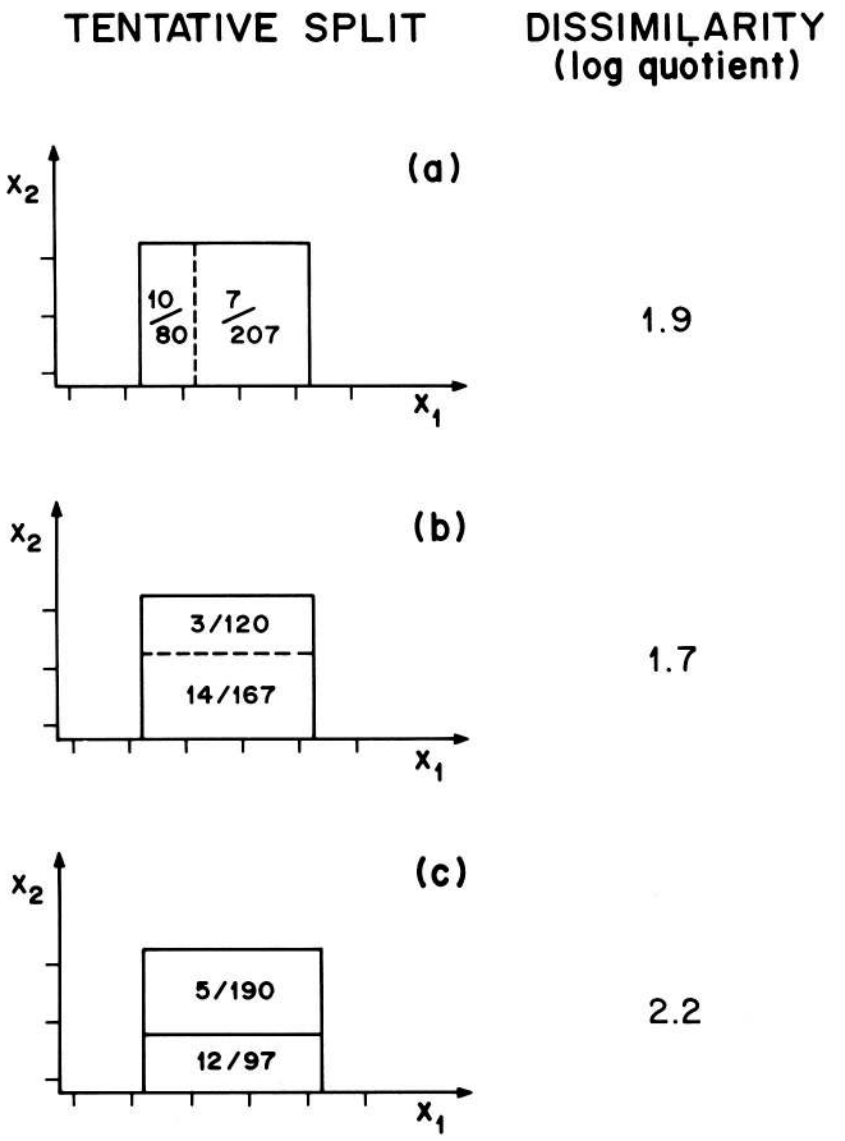

Figure 9. A simplified view of the optimal discrimination criterion (without consideration of error). For various ways of splitting the data, the probability quotient or information content is computed to find the best one (see text). Here the most discriminating choice is (c).

utility. In other words, the best occurs where u_1/u_2 is maximum (if u_1 is the larger of the two utilities). Equivalently, we can maximize the dissimilarity measure $d = |\log u_1 - \log u_2|$. Notice that $d = 0$ means equal utilities. In Figure 9 the largest value of d is 2.2, so the corresponding dichotomy (c) would be retained.

Accounting for uncertainty. While this dissimilarity measure d is a good one, we can improve it. If u_1 and u_2 are the two probabilities for a tentative dichotomy, and e_1 and e_2 their *error factors*, then $d = | \log u_1 - \log u_2 | - \log(e_1 e_2)$. Notice that d can be negative if the error factors are greater than 1. The most obvious source of error is related to the sample size n, although there are other sources of random and systematic error (Rendell, 1983a). To split a rectangle, compute d for a number of choices. If the largest d is positive, retain the corresponding split. Repeat the process until additional refinement is unwarranted by the data (until $d \leq 0$). Larger values of d mean more assured dissimilarity. A formal derivation of our criterion d is based on statistical analysis (Rendell, 1981). Rendell (1977) introduced this approach in PLS1 to discover locally optimal evaluation functions.

Similar measures. Our probability-based dissimilarity d is much like the information-theoretic measure in Quinlan's ID3 (1983, pp. 467 ff.); see also (Watanabe, 1969, pp. 27 ff.). One can verify that, in terms of $u = g/n$, the information criterion for dichotomies $d' = -u \log u - (1-u) \log (1-u) - [1/n] \Sigma [g_i \log u_i + (n_i - g_i) \log (1 - u_i)]$. Table 2 shows some comparisons between d and d'. Notice that the two measures are quite similar, except that d accounts for errors, which is especially important for small or non-random samples. Either measure can be considered a component in the overall credibility μ of an hypothesis.

Table 2. Values of similar criteria for intelligent discrimination. Shown are the inductive criterion d for PLS1 (with and without correction for error e), and the criterion d' for ID3 (larger is better throughout, and a zero or negative value means no splitting is justified). Splitting governed by utility dissimilarity d or d' lessens our ignorance of the utility or probability of success in a task. In (a) the values correspond to Figure 9. Notice that d and d' produce the same result. In (b) the values listed are variations of the first entry of group (a). Notice that the last entry of (b) beats any entry in (a); however the bias of the representation language prohibits this case. The last group (c) shows the effect of a small sample of 20 data. The pure information criterion d' fails to account for the lack of confidence associated with a small sample, while d naturally covers all cases.

	g_1	n_1	g_2	n_2	d (no e)	d	d'
	10	80	7	207	1.9	1.5	0.019
(a)	3	120	14	167	1.7	1.4	0.012
	12	97	5	190	2.2	1.9	0.026
	1	80	16	207	2.6	2.3	0.014
(b)	5	80	12	207	0.1	−0.3	0.000
	10	80	7	207	1.9	1.5	0.019
	15	80	2	207	4.3	3.9	0.074
	2	8	6	12	1.0	−0.2	0.046
(c)	4	8	4	12	0.6	−0.6	0.020
	6	8	2	12	2.2	1.0	0.256

Summary. Earlier we saw that the probabilistic information for a sub-hypothesis or region is concisely represented, and now we see the same for a set of regions. When used for iterative induction, our scheme for *intelligent discrimination* minimizes the number of sub-hypotheses or regions, since splitting occurs only when justified. Despite the compression, the critical information remains. In terms of our credibility μ for hypotheses, only those hypotheses (region sets) arise which are believed discriminative.

Intelligent discrimination appears in slightly different forms in two learning systems which may seem unrelated on first sight (PLS1 and ID3). Their statistical and information criteria are largely independent of the particular algorithm used for discrimination; in fact Rendell (1985a) has used d for generalization and reorganization of hypotheses. In the following section we examine other commonalities in systems for induction.

2.3.2 A unifying view of systems for selective induction

Constructing decision trees is selective induction. Some learning systems represent a concept as a decision tree (Quinlan, 1983). A system which builds a decision tree chooses one of k attributes x_i ($i \leq k$), and the system labels the resulting branches with values of x_i (Watanabe, 1969). Every level in the tree corresponds to a different attribute, so the depth of the tree is at most k, although we just saw that splits make sense only if the value of the discrimination criterion d is large enough.

Imagine the construction of a decision tree from an initial node representing the whole universe. Suppose that attribute x_i is selected and that x_i has c values. Depending on the particular algorithm, as many as c branches result (note that we do not need to discriminate all values, especially if subsets of them are similar according to our criterion d). Next, some other attribute x_j is selected and a new level of the tree is formed; this process continues until the attributes are exhausted or until d no longer discriminates training examples.

What is the relationship between a decision tree and a region set? Both represent a single concept. But more specifically, if the attributes are interval or binary, then branches at one level of tree correspond to partitions in one dimension of feature space. Branching on every attribute value corresponds to exhaustive partitioning in one dimension. New levels in the tree correspond to partitioning in new dimensions. A decision tree retains the discriminatory ranking of attributes; a region set retains probability and error. Neither of these differences is inherent, but some others are. A region set does not require that an attribute be considered only once (this permits more refined discrimination; see Figure 8). A region set facilitates incremental learning.

Comparing learning systems is important. As well as comparing representations and algorithmic components, we can relate overall aspects of performance such as speed.

ID3 and PLS1 are both efficient (**O**(knm), where k is the number of attributes, n is the number of data, and m is the ultimate number of classes). In contrast, some algorithms are relatively slow, as O'Rorke (1982) has shown. The quantification we began in Section 1.4 to measure the conceptual knowledge acquisition can be extended to allow a cost/benefit analysis. This should be done cautiously, since there are many factors – but sometimes the differences are so marked that any doubts disappear.

Consider the advantages of comparing learning systems. If we juxtapose systems, commonalities and differences in their elements become obvious. By comparing ID3 and PLS1, we have seen striking similarities in their discrimination criteria, and useful differences in their concept representation. We should ask why these similarities arise, which components are responsible, and whether improvement is possible. In what ways is Quinlan's (1986) recent development of ID3 like the error treatment of Rendell's (1977) PLS1 (see Table 2)? Comparison of systems may also suggest improvements at points of dissimilarity. Comparitive study can increase our ability to understand, to apply, and to develop representations and methods.

A region set is a node in a probabilistic version space. Continuing with our effort to unify learning systems for selective induction, let us consider how we might construct version spaces to manage noise. As we have been seen earlier in this paper, version spaces are desirable because they economically represent whole sets of hypotheses as subspace boundaries (see Figure 1).

The common conception of version spaces is that they are hampered by their apparent inability to accommodate errors (Mitchell, 1978; Quinlan, 1983). But let us consider this question more carefully. In a normal version space, the nodes in the lattice are binary hypotheses (Figure 1). Indirectly, Figure 3 suggests a probabilistic version space, since it shows a sub-hypothesis r paired with potential information u = Pr(C | r) about the probability that r is part of the desired concept C. In essence, Figure 3 depicts a region (r, u, ...) as a single node. But a region is a component of the utility function u(**x**) expressed as a region set **R** = {R}. It is this function u which ultimately represents C. Hence, we need a version space in which the nodes are region sets, as shown in Figure 10, a derivative of Figure 3, in which the training instances have been converted to probabilities.

The region set **R** is precisely the hypothesis representation which PLS1/2 manipulates in its probabilistic concept formation. The structure imposed, however, is somewhat more complex than a simple lattice. While the generalization and specialization operators may transform one hypothesis into another by following branches in a lattice, reorganization operators can hop to a different part of the lattice. Further, associated with descriptions r of sub-hypotheses are probabilistic quantities such as u and e, and these additionally complicate the space. Revision operators altering u and e also jump to another part of the lattice. See (Rendell, 1985a).

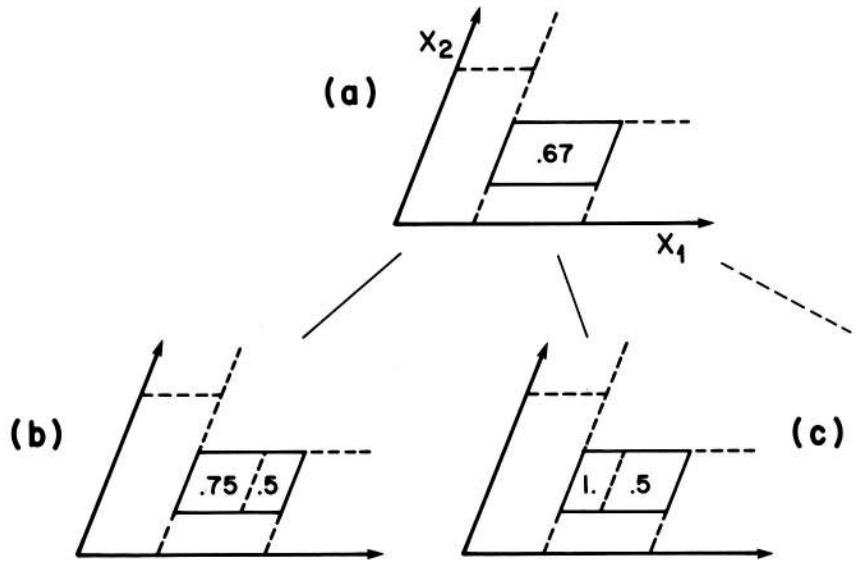

Figure 10. A portion of a probabilistic version space for a concept C. Instead of being a rigid hypothesis, each node in the lattice is a set of flexible sub-hypotheses (i.e. a set of regions). A sub-hypothesis expresses the probability u that a rectangle r implies C. The whole set of sub-hypotheses (region set) expresses the degree or probability of class membership as a function u of attributes.

Data-driven operations make PLS1/2 more efficient than Figures 3 and 10 suggest. See (Rendell, 1986).

Inducing utility functions is conceptual clustering. As another topic in our discussion of inductive schemes, let us consider unsupervised learning, or clustering. The clustering problem is the problem of aggregating objects when a teacher does not present training examples as members of specific classes (Watanabe, 1972). Since induction is arbitrary without some imposed criterion, clustering must employ something other than just the objects. The bias or model chosen often takes the form of a feature space description, and algorithms are frequently based on Euclidean distance in the space. Since features are arbitrary, however, the proper definition of distance is not generally known. Hence "external criteria" are sometimes imposed (Anderberg, 1973, pp. 194 ff.). Rendell (1977) suggested the external criterion d based on task utility u, which is like another feature except that utility is not simply the property of a single object, but rather of the the whole data environment. Another kind of external criterion is based on the description language. In fact we have seen that as soon as we describe an object, we begin to constrain the classes which can be expressed. We may limit not only the object description, but also the concept description (as in Figure 3). Thus we return to one of our earlier themes: powerful induction is both data- and model-driven. When clustering is constrained

not only by the description of individual objects, but also by their relationships to other objects, and by a restricted set of concept descriptions, it is sometimes called *conceptual clustering* (Michalski and Stepp, 1983).

How does this relate to our situation? Since the formation of concepts as region sets creates probability classes, this induction is clustering (see Figures 8 & 9). Since this clustering depends both on the environment (utility), and on limited concept forms (rectangles), it is conceptual clustering (Rendell, 1977). Except for the fact that ID3 produces binary classes, it could also be doing conceptual clustering: like PLS1, ID3 is guided by probabilistic utility, and the hypotheses of ID3 are biased by implicit disjunction.

Unsupervised learning is like supervised learning. As a closing remark, we begin by noticing once again that concept formation is the creation of a utility function u. The data for u are given in one of two ways: the data are positive and negative training examples (then u is 1 or 0); or the data may be noisy (then u is in the range [0, 1]). In either case the learning is supervised. In certain other cases, though, the data may not be so clearly distinguished. For example, when the data result from guided search, it is not always obvious how to assign values to u (see Rendell, 1983a). Here the value of u, even for the data, becomes partially subjective. Now, suppose that we remove all traces of objective evidence. Then we have the clustering problem, or unsupervised learning, and the learner must supply u entirely subjectively (e.g. by analogy). This distinction between supervised and unsupervised learning is not quite the usual one.

Summary. We have seen that there are many commonalities among inductive systems which are superficially distinct. Common aspects include concise representation, maximal discrimination, and concerted use of model and data. When we describe, analyze, or compare learning systems, we should carefully distinguish their *abstract* elements. Components such as the inductive criterion d are not bound to particular representations or algorithms; nor are external manifestations the ultimate conceptions (Kanal & Chandrasekaran, 1972; Holte, 1986). We should focus on the essence: precisely what is responsible for the efficacy and efficiency of powerful methods? We have begun to answer some of these questions, but many more remain. We also reiterate that careful comparison of various systems will continue to consolidate and sharpen our understanding and implementation. Despite various ideas such as utility functions, probabilistic version spaces, and information-based clustering and discrimination, essential principles are still formative.

2.3.3 Some emerging principles for power

Ideally, an effective and efficient induction method should be incremental, it should converge quickly to accurate or optimal concepts, and the method should permit

concise expression of concepts. Let us summarize some of the more important means for power.

Synergic belief or credibility. In situations of real-world complexity, a desired concept emerges only after many sources of knowledge have been polled. This may be a complex process involving several stages and hypotheses, and a powerful inductive system must accommodate and unify the various sources and elements. Integrated hypothesis assessment may be formulated using belief or credibility μ. μ may be rigid and binary, but the real-world demands refined comparison, so μ is multi-valued and probabilistic. Ideally the probability would be objective, a product of extensive observation – but complexity is so extreme that only small samples can be used, and only a few hypotheses can be tested. Hence μ becomes a largely subjective probability constraining hypotheses *formation.* (See Section 1.2 for a general treatment, and Section 2.2 ff. for details.)

Multiple and flexible constraints or biases. Hypothesis constraint may be achieved by language limitation and other methods. The constraint or bias speeds the inductive process but may also vitiate it, so the ideal learning system Λ should select its own bias. The selection should be based on measurement or self-assessment, and this is facilitated using our probabilistic representation. If sub-hypotheses (regions R) cannot discriminate utilities properly, then the bias producing these R must be inadequate. (See Sections 1.3, 1.4 and 2.2 ff.)

Use of probability to manage and even to exploit uncertainty. We have seen the use of probability on at least two levels: the credibility μ assesses the quality of an hypothesis H, and the utility u assesses the quality of an object **x**. Associated with each of μ and u is its second order probability or confidence. If our learning system Λ is unsure of μ, it retains several hypotheses H with high μ values. Λ manages uncertainty in u by recording its error e; moreover Λ discriminates sub-hypothesis r *only if* u and e warrant it. Hence Λ converts the undesirable lack of confidence into a desirable cognitive economy (Sections 2.2 & 2.3).

Full but controlled use of every datum to induce and to assess hypotheses. These values of probability u and error e are computed using every single object **x** encountered. No one datum can errantly overwhelm the system, yet each one updates Λ's knowledge. The added knowledge improves the utility function u(**x**), and the same information can improve the assessment μ of the function u. The utility function u *is* the hypothetical concept. (See Section 1.2 about the use of data in a definition of μ, 1.3.2 about induction as the discovery of credible utility functions, and 2.2 and 2.3 about the use of data for the discrimination and clustering of utility.)

Intelligent discrimination, implicit disjunction, and mutual data support. We have also seen that a powerful system Λ can maximize discrimination and minimize storage by extracting appropriate information from the data (Section 2.3.1). In selective induction, the data are compressed into concepts by implicit disjunction (Figure 2 and Section 2.1). Λ may use scarce information *simultaneously* to compress data, manage noise, to improve accuracy, and to form concepts. This *mutual data support* is important.

Representation of whole sets of hypotheses using boundaries. Mutual data support involves cognitive economy for formation of sub-hypotheses. This same sort of economy appears at the higher level of complete hypothesis formation. Mitchell's candidate elimination for binary version spaces is economical because concise boundaries represent whole sets of hypotheses (boundaries gradually converge). Using a sort of probabilistic version space, PLS1 is similarly economical because tentative boundaries represent the restricted set of partially confirmed hypotheses (boundaries provisionally converge, with increasing assurance; see Figures 1, 3 & 10, and Section 2.2 ff.).

Concerted use of data and model. Running throughout this paper is another important theme: effective and efficient concept formation in a powerful system Λ relies on synergic use of both data and model, i.e. of both observation and bias. This idea appears in many forms, for example in conceptual clustering. (The general notions are examined beginning in Section 1.3, and specific forms appear later, e.g. in Section 2.3.2.)

Mediating structures. Successful inductive systems tend to employ information structures which *mediate* data objects and the ultimate knowledge form. One of these is the region set **R** which records and improves a concise form of the concept or utility function. Learning systems often attempt to improve the hypothesis H itself, whereas our scheme Λ acquires knowledge efficiently in **R**, and compresses it only temporarily for task performance. In other words, Λ does not directly search the original space of hypotheses for a credible H, but rather searches an easier space {**R**} for components (sub-hypotheses) from which H is constructed. This notion of mediating structures is important; see (Rendell, 1985a; Holte, 1986).

Invariance and extension. Throughout this present paper we have looked beyond surface differences in learning systems to extract fundamental ingredients for powerful induction (see also Holte, 1986). By developing and unifying notions such as cognitive economy from cognition and computational complexity from computer science, we may form a concise and powerful theory for practical induction. Many, if not all of these notions, extend to the most difficult cases of structure formation (Rendell 1985b).

3. Conclusions

Practical problems of induction involve two basic issues: efficacy and efficiency (power). The first major part of this paper was an extensive examination of a general framework to facilitate this power. We considered means to represent, to create, and to test hypotheses as credible versions of a desired concept. Various sources of knowledge contribute to an overall measure of the quality of an hypothesis, its credibility μ. μ is largely a belief, a subjective probability, although objective measures may also be used sparingly. We considered components and examples of μ.

We developed a measure for the difficulty Γ of an induction problem, and we saw that Γ also measures the bias strength or degree of constraint placed on the learning system. Since the original problem difficulty remains constant, we can compute the amount of induction left to the system by subtracting the effect of various biases. The remainder is the knowledge the system attains on its own. Such computations give us an idea of such things as the inherent difficulty of a problem, and the quality of a learning system.

Our general analysis of induction seems to capture the basic requirements for powerful induction, such as the capability for noise-resilient incremental learning. Importantly, the problem of induction *is* the problem of discovering a function which expresses the utility of an object relative to the purpose of induction. This utility function u *is* the concept. u may be Boolean (for rigid, binary concepts), or u may be multi-valued (for flexible probabilistic concepts). The regularity of the utility function determines the difficulty of the induction.

This brings us to the second major part of this paper, which was an extensive study of induction in cases where the utility functions are quite regular (this "selective induction" requires implicit disjunction). We examined several learning approaches and systems for selective induction, and we found that there are more similarities than commonly supposed. These include maximal discrimination of utility and means for speed and cognitive economy. We also suggested some improvements such as probabilistic version spaces.

As a result of our study we have extracted some incipient principles for powerful induction. These include a systematic method for the efficient and effective creation of hypotheses using optimal sub-hypotheses; other schemes for inductive power are summarized in the previous section. Perhaps the most important contribution of this paper is some schemes for attacking the general problem of practical induction, schemes which may extend to the most difficult cases. These abstractions look beyond the superficial differences among learning systems: toward concise, invariant, and fundamental laws.

Acknowledgements

Some of the ideas developed in this paper became clearer after discussions this past summer at the Machine Learning Workshop, the Workshop on Uncertainty in AI, and IJCAI. In particular, Ranan Banerji, Peter Cheeseman, Rob Holte, and Ray Solomonoff have had an effect on this paper. At Illinois, several other friends and colleagues have provided useful comments on various drafts, and have contributed stimulating ideas during discussions. I would especially like to thank Raymond Board, Brian Falkenhainer, Peter Haddawy, Chris Matheus, Doug Medin, Ryszard Michalski, Igor Mozetic, and Dave Tcheng. Pat Langley offered many helpful suggestions, which I very much appreciate.

This work was supported primarily by an operating grant from the Natural Sciences and Engineering Research Council of Canada, and in part by the Advanced Research Projects Agency of the Department of Defence under grant N00014-K-85-0878.

References

Anderberg, M.R. (1973). *Cluster analysis for applications*. New York: Academic Press.

Angluin, D., & Smith, C.H. (1983). Inductive inference: Theory and methods. *ACM Computing Surveys 15*, 237–269.

Banerji, R.B. (1985). The logic of learning: A basis for pattern recognition and for improvement of performance. *Advances in Computers, 24*, 177–216.

Bruner, J.S., Goodnow, J.J., & Austin, G.A. (1956). *A study of thinking*. New York: Wiley.

Cheeseman, P. (1985). In defense of probability. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, (pp. 1003–1009). Los Angeles: Morgan-Kaufmann.

Christensen, R. (1964). *Foundations of inductive reasoning*. Berkeley: Entropy, Ltd.

Dietterich, T.G., London, B., Clarkson, K., & Dromey, G. (1982). Learning and inductive inference. In P.R. Cohen & E.A. Feigenbaum (Ed.), *The handbook of artificial intelligence*. Los Altos: Kaufmann.

Duda, R.O., & Hart, P.E. (1973). *Pattern classification and scene analysis*. New York: Wiley.

Easterlin, J.D., & Langley, P. (1985). A framework for concept formation. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, (pp. 267–271). Irvine, CA.

Erman, L.D., Hayes-Roth, F., Lesser, V.R., & Reddy, D.R. (1980). The HEARSAY-II speech understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys, 12*, 213–253.

Glass, A.L., & Holyoak, K.J. (1975). Alternative conceptions of semantic memory, *Cognition, 3*, 313–339.

Holland, J.H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

Holte, R.C. (1986). Alternative information structures in incremental learning systems. *Proceedings of the European Working Session on Learning*.

Hunt, E.B., Marin, J., & Stone, P.J. (1966). *Experiments in induction*. New York: Academic Press.

Kanal, L., & Chandrasekaran B. (1972). On linguistic, statistical and mixed models for pattern recognition. In S. Watanabe (Ed.), *Frontiers of pattern recognition* (pp. 163–192). New York: Academic Press.

Koestler, A. (1964). *The act of creation: A study of the conscious and unconscious in science and art*. New York: Macmillan (hard cover). (Some other editions omit the important Book Two).

Langley, P. (1985). Learning search strategies through discrimination. *International Journal of Man-Machine Studies, 18*, 513–541.

Lenat, D.B. (1983). The role of heuristics in learning by discovery: Three case studies. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (pp. 243–306). Palo Alto: Tioga.

Matheus, C.J. (1985). Views on concept representation and implications for machine learning. Unpublished manuscript.

Medin, D.L., & Smith, E.E. (1984). Concepts and concept formation. *Annual Review of Psychology, 35*, 113–138.

Medin, D.L., & Wattenmaker, W.D. (1985). Category cohesiveness, theories, and cognitive archeology. In U. Neisser (Ed.), *Concepts reconsidered: The ecological and intellectual bases of categories.*

Michalski, R.S. (1983). A theory and methodology of inductive learning. *Artificial Intelligence, 20*(2), 111–161. Reprinted in R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (pp. 83–134). Palo Alto: Tioga.

Michalski, R.S., & Chilausky, R.L. (1980). Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *International Journal Policy Analysis and Information Systems, 4*, 125–161.

Michalski, R.S., & Stepp, R.E. (1983). Learning from observation: Conceptual clustering, In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (pp. 331–363). Palo Alto: Tioga.

Michie, D. (1977). A theory of advice. In E.W. Elcock, & D. Michie (Eds.), *Machine intelligence, 8* (pp. 151–168). American Elsevier.

Mitchell, T.M. (1978). *Version spaces: An approach to concept learning.* Unpublished doctoral dissertation, Stanford University, Stanford, California.

Mitchell, T.M. (1980). *The need for biases in learning generalizations*, (Technical Report No. CBM-TR-117). New Brunswick: Rutgers University, Dept. of Computer Science.

Mitchell, T.M. (1982). Generalization as search. *Artificial Intelligence, 21*, 203–226.

Murphy, G.L., & Medin, D.L. (1985). The role of theories in conceptual coherence. *Psychological Review, 92.*

O'Rorke, P. (1982). *A comparative study of inductive learning systems AQ11P and ID-3* using a chess endgame test problem (Technical Report No. UIUCDCS-F-82–899 and ISG 82–2). Urbana-Champaign: University of Illinois, Dept. of Computer Science.

Quinlan, J.R. (1983). Learning efficient classification procedures and their application to chess end games. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds), *Machine learning: An artificial intelligence approach* (pp. 463–482). Palo Alto: Tioga.

Quinlan, J.R. (1986). Induction of decision trees. *Machine Learning 1*, 81–106.

Rendell, L.A. (1977). *A locally optimal solution of the fifteen puzzle produced by an automatic* evaluation function generator (Technical Report No. CS-77–36). Waterloo, Ontario, Canada: University of Waterloo, Dept of Computer Science.

Rendell, L.A. (1981). *An adaptive plan for state-space problems.* Unpublished doctoral dissertation (Technical Report No. CS-81–13), University of Waterloo, Dept. of Computer Science, Waterloo, Ontario, Canada.

Rendell, L.A. (1983a). A new basis for state-space learning systems and a successful implementation, *Artificial Intelligence, 20*, 369–392.

Rendell, L.A. (1983b). *Conceptual knowledge acquisition in search* (Technical Report No. CIS-83–15). Guelph, Ontario, Canada: University of Guelph, Dept. of Computing and Information Science. Also L. Bolc (Ed.). *Computational models of learning*, Springer-Verlag).

Rendell, L.A. (1985a). Genetic plans and the probabilistic learning system: Synthesis and results, *Proceedings International Conference on Genetic Algorithms and their Applications* (pp. 60–73).

Rendell, L.A. (1985b). Substantial constructive induction using layered information compression: Tractable feature formation in search, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 650–658).

Rendell, L.A. (1986). Uncertain and incremental concept learning: Efficiency through model and data synergy. Unpublished manuscript.

Ritchie, G.D., & Hanna, F.K. (1984). AM: A case study in AI methodology. *Artificial Intelligence, 23*, 249–268.

Samuel, A.L. (1967). Some studies in machine learning using the game of checkers II – recent progress. *IBM Journal of Research & Development, 11*, 601–617.

Tou, T.T., & Gonzalez, R.C. (1974). *Pattern recognition principles*, Reading: Addison-Wesley.

Utgoff, P.E., & Mitchell, T.M. (1982). Acquisition of appropriate bias for inductive concept learning. *Proceedings of the National Conference on Artificial Intelligence*, 414–417.

Watanabe, S. (1969). *Knowing and guessing: A formal and quantitative study*, New York: Wiley.

Watanabe, S. (1972). Pattern recognition as information compression. In Watanabe, S. (Ed.). *Frontiers of pattern recognition* (pp. 561–567). New York: Academic Press.

Winston, P.H. (1984). *Artificial intelligence* (2nd ed.). Reading: Addison Wesley.

Wise, B.P., & Henrion, M. (1985). A framework for comparing uncertain inference systems to probability. *Proceedings of Uncertainty and Probability in Artificial Intelligence Workshop* (pp. 99–108).