Journal of the Korean
OR/ MS Society
Vol. 19, No. 2, August 1994

An Introduction of Machine Learning Theory to Business Decisions

Hyunsoc Kim*

Abstract

In this paper we introduce machine learning theory to business domains for business decisions. First, we review machine learning in general. We give a new look on a previous learning framework, version space approach, and we introduce PAC (probably approximately correct) learning paradigm which has been developed recently. We illustrate major results of PAC learning with business examples. And then, we give a theoretical analysis of decision tree induction algorithms by the framework of PAC learning. Finally, we will discuss implications of learning theory to business domains.

1. Introduction

There are two major approaches to studying learning. Cognitive scientists try to develop theories and models of learning observable in humans and other animals. Researchers in Artificial Intelligence develop theories and models of any type of learning. Those theories and models do not necessarily involve living organisms. Our focus here is on learning as studied by AI researchers.

Knowledge acquisition for expert systems is a time consuming process that has bedeviled many attempts at fielding expert systems applications. Johnson [12] coind the term "paradox of expertise" which describes a phenomenon that the more an expert knows, the less able is he or she articulate that knowledge. Since expert often find it hard to articulate their exeprtise, many researchers are trying to develop alternative knowledge acquisition methods such as machine learning.

In this paper we introduce machine learning theory to business domains. In Section 2, we review machine learning in general. We will give a new look on a previous learning framework,

* Department of Management Information Systems, Kookmin University

and introduce PAC (probably approximately correct) learning paradigm in Section 3. We also illustrate major results of PAC learning with examples. In Section 4, we give a theoretical analysis of decision tree induction algorithms by the framework of PAC learning. Finally, in Section 5, we will discuss implications of learning theory to business domains.

2. Machine Learning

In this section we summarize different views of learning and learning strategies. We also discuss differences of terms in machine learning.

2.1 Views of Learning

There are several different point of views of learning. Cohen and Feigenbaum [6] lists four views of learning.

(1) any process by which a system improves its performance.
(2) the acquisition of explicit knowledge
(3) skill acquisition
(4) theory formation and discovery

The first view, improving performance on a given problem, is the most studied form of learning. Valiant [42] describes learning as "the phenomenon of knowledge acquisition in the absense of explicit programming." This view of learning grew out of research in problem solving.

The second view, the acquisition of explicit knowledge is a more limited view of task performance. For example, most expert systems have an explicit collection of rules. Neural nets learn but the knowledge is not explicit.

Skill acquisition refers to the phenomenon whereby one becomes more proficient at a task with practice. Finally, theory formation and discovery views learning from the process of scientific discovery of principles and theories.

Our focus in this paper is on the first two view of learning.

2.2 Learning strategies

There are several basic learning paradigms or learning strategies. Of particular interests are [6]: rote learning, learning by being told (advice taking), learning from examples (induction), and learning by analogy. A brief description of the four learning situations follows. Suppose that a learning system is embedded in an environment of interest and a knowledge base is used by the performance element. Here the knowledge base is a collection of knowledge and the performance element is a system performing tasks by using the knowledge base.

1. Rote learning : The environment supplies knowledge in a form that can be used directly by the performance element. The learning system just needs to memorize the knowledge for later use. Though rote learning is a rudimentary type of learning, this is widely used in daily human life.
2. Learning by being told : Here the environment gives vague, general purpose knowledge or advice. A learning system must transform this high-level knowledge into a form that can be used readily by the performance element. Davis's [7] TEIRESIAS is an example of this type of learning.
3. Learning from examples : In learning from examples, examples are given to the learning system. The system generalizes these examples to find higher level rules that can be used by the performance element. This type of learning has a long history under the name of "induction" and is a powerful method of acquiring knowledge.
4. Learning by analogy : If a system has available to it a knowledge base for a related performance task, it may be able to improve its own performance by recognizing analogies and transferring the relevant knowledge from the other knowledge base.

Some researchers add more types of learning to the above four types of learning. For instance, Shaw et al. [37] list two more types of learning, learning by competition, and learning from observation and discovery.

Learning situation can also be categorized by settings where learning takes place. Learning can take place in the following two broad settings :

(1) 'supervised' learning : a teacher (or an all-knowing oracle) is present. The presence of teacher removes ambiguities from the training set. The machine can then learn much more rapidly and efficiently.

(2) 'unsupervised' learning : the learning system has no instructor but must acquire knowledge on its own. The training set may be full of ambiguities which the learning algorithm must resolve on its own. If the training set is not sufficiently large, the learning algorithm may fail to

perform at an acceptable level.

The above mentioned learning strategies reflect a decreasing reliance of supervision and an increasing complexity of the inference process. For example, in rote learning, the teacher directly supplies information. No inference is needed. Learning by analogy involves little supervision but requires a complex inference capability.

Within each general strategy, we employ different inference mechanisms to varying degrees. The main inference mechanisms are deduction and induction. Deduction moves from general truths to specific cases whereas induction moves from specific cases to generalizations.

Deductive information processing is "truth preserving." All 'truths' classified by the deduced information are implied by the initial information. Hence, new information preserves the facts contained in old information. Deriving specific facts from general rules or developing new rules from old ones are deductive procedures.

Inductive information processing is 'falsity preserving.' The induced information correctly categorizes all fallacies contained in the initial knowledge. Using raw data or examples to establish raws, rules or general patterns, are examples of inductive procedures.

Inductive learning is generally considered synonymous with learning from examples. However, Angluin and Smith [2] distinguish inductive inference (i. e. , inductive learning) from learning from examples. They say that work in artificial intelligence (i. e. , learning from examples) is more concerned with cognitive modeling than the work in inductive inference, and less concerned with formal properties such as convergence in the limit or computational efficiency. A learning algorithm is said to learn a concept in the limit if, after some finite number of examples, the learner's hypothesis is correct, and thereafter all the learner's hypotheses remain correct [18]. Convergence in the limit does capture notion that the learner will eventually discard any false hypotheses, and that in finite time this progression will ultimately converge to a fixed, correct rule. The computational complexity of a learning algorithm is defined if and only if the algorithm converges [2]. Computational efficiency can be considered as an additional important property of an inductive learning algorithm in practice.

3. Learning Theory

There has been an explosive growth in the theory of learning quite recently. Mitchell's version space [25], Valiant's PAC (Probably Approximately Correct) learning [1, 42], and Haussler's researches [9, 10, 11] are important building blocks for the learning theory. In general, learning

theory considers three aspects of the learning process : concept accuracy, storage efficiency, and computational efficiency. An acceptable learning algorithm must operate within reasonable storage and time limitations to produce an acceptably accurate concept. 'Reasonable' usually means some polynomially bounded measure. Concept accuracy shows 'how well' the system learns. This is the percentage of instances correctly classified by the learned concept. This measure is well suited for description or classification tasks. However, it may not be appropriate for a pattern-matching task.

Storage efficiency indicates 'how costly' is it for the system to learn. Memory is a resource that system developers must manage well. Thus, superior memory management for a given task demonstrates improved performance.

Computational efficiency reveals 'how long' the system takes to learn. A desirable property for any application is having the computational process by which the machine learns be a small number of steps.

In this section we look at the main stream learning theory in terms of these three important considerations.

3.1 Mitchell's Version Space

Mitchell [25] gives an elegant framework for viewing the process of learning from examples and illustrates this framework by analyzing the process of learning simple conjunctive concepts. We start with basic definitions and terminology used in his framework.

Definition 3. 0 :

1. The instance space : The instance space is the space of all objects of interest. Each instance of a concept can be expressed in a feature-based or attribute-based form. Attribute-based instance spaces can be defined by the values of a fixed set of attributes, not all of which are necessarily relevant. Feature (or structure)-based instance spaces can be defined by allowing each instance to include several objects, each with its own attributes, and allowing binary relations that define a structure between objects. For example, define an instance space consisting of the following attributes and binary relations [10]. Permissible values appear in parentheses.

Attributes : . size (small, medium, large)
. shape (convex, nonconvex)
Binary relations :
. distance-between (touching, nontouching)
. relative-position (on-top-of, under)

Then (size=small, shape=convex) is an example of an instance of an attribute-based instance space. The following instance is an example of a structure-based instance space.

(size=small, shape=convex)
(under, touching) ↑ ↓ (on-top-of, touching)
(size=large, shape=nonconvex)

2. Hypothesis space : The hypothesis space is the space of all plausible hypotheses. It is often called the rule space.

3. Inductive bias : A mechanism whereby the space of hypotheses is restricted or whereby some hypotheses are preferred, a priori, over others reflects the inductive bias.

4. Conjunctive concepts : Concepts described by logical expressions involving only conjunctions (i. e. , AND operations) are called conjunctive concepts.

5. Disjunctive concepts : Concepts described by logical expressions involving only disjunction (i. e. , Inclusive OR operations) are called disjunctive concepts.

6. Target concept : The target concept is the true concept. A learning system tries to find the target concept.

Mitchell's framework of learning can be described as follows. Let us assume that we are trying to learn some unknown target concept, f, defined on the instance space. This target concept can be any subset of the instance space.

This may or may not be a conjunctive concept. Assume we have a set of examples of this target concept. Each example is generated by "sampling with replacement", and is one of following two cases.

Case 1. An instance satisfying the target concept.

Case 2. An instance not satisfying the target concept.

An example in Case 1 is called a "positive example", and an example in Case 2 is called a "negative example". We label each example accordingly. We call these labelled examples a sample, S, of the target concept.

We assume a hypothesis space, H, restricted to only conjunctive concepts. This is an inductive bias. Then the task is to produce a conjunctive concept that is consistent with the sample or to detect when no conjunctive concept is consistent with the sample. By "consistent" we require that a concept contains all instances of positive examples and no negative example.

The version space is the set of all hypotheses $h \in H$ that are consistent with the sample. Since the version space depends on the hypothesis space, we denote the version space with respect to the hypothesis space H. The version space is empty in the case that no hypothesis in H is consistent with the sample.

Mitchell shows that the learning task (and related tasks) of producing a conjunctive concept consistent with the sample can be solved by keeping track of only two subsets of the version space — the set of the most specific hypotheses and the set of the most general hypotheses. These sets are updated accordingly as new examples are given.

Here we consider a finite instance space, and we assume no examples are in contradiction each other. There are two cases for the target concept :

Case 1 : The target concept $f \in H$

Case 2 : The target concept $f \notin H$

For Case 1, the version space reduces until it contains only the target concept f, as examples are added to the version space. For Case 2, the version space reduces until it becomes the empty set. Note that for both cases, the version space reduces to an informative terminal state which can tell the result of the learning task. If we stop before one of the terminal states is produced, then the learning task is incomplete and the current result is not as useful.

We say that the version space is exhausted with respect to H (we abbreviate this as "w. r. t. H") if the version space is reduced to one of the above terminal states (i. e. , is reduced to either the target concept f or an empty set).

Consider the situation that the version space contains only one hypothesis h, where h is not the target concept. If the target concept f is not an element of the hypothesis space H, then this situation may occur. For this situation, we assume that it is always possible to generate a new example which eliminates h from the version space. The version space will be empty and then be exhausted.

Mitchell's approach to inductive learning is to sample until the version space is exhausted. Stopping short of an exhausted version space leaves one with incomplete learning. However, the two subsets of version space bound the space where the target concept may exist.

3.2 Problems of Mitchell's Framework

There are two practical problems with Mitchell's approach. The first is that it may require too many examples to exhaust the version space [9].

The other problem is that even if we monitor only the two sets, the set of the most specific hypotheses and the set of the most general hypotheses, the storage needed can still become exponentially large as we build up examples [9].

These problems with the version space approach are overcome by incorporating probablistic

ideas [42]. In the following we give a concise presentation of this new idea by the help of Haussler [9]'s articulation. Here we will not require the complete exhaustion of the version space. Instead, we will require that a version space is "probably almost exhausted". (This term will be formally defined later.) This idea will do away with the first problem.

To handle the second problem, we will not try to remember the exact version space. Instead, we will require that any hypothesis from an "almost exhausted" version space will accurately approximate the target concept.

Hence, we replace Mitchell's idea of remembering all consistent hypotheses by a more elegant idea of drawing enough examples needed for a "probably almost exhaustion" of the version space and then finding an hypothesis (or hypotheses) consistent with these examples. Following Definition 3.1 gives a formal definition of ε-exhaustion.

Definition 3.1 : [9] Given a hypothesis space H, a target concept f, a sequence of examples S of f, and an error tolerance ε, where $0<\varepsilon<1$, the version space of S (w.r.t. H) is ε-exhausted (w.r.t. f) if it does not contain any hypothesis that has error ("error" will be formally defined in the next subsection) more than with respect to f.

3.3 Efficient PAC Learning

Now we introduce a formal definition of Probably Approximately Correct (PAC) learning based on the idea of an "almost exhausted" version space defined in the previous subsection. The concept of computational efficiency is also very important for a learning algorithm to be practical for larger problems. The PAC learning paradigm was introduced by Valiant in 1984. This model requires that a polynomial bounded algorithm identify a concept using a random sample, whose size is polynomially bounded, such that a learned concept has a high probability of being close to the true concept. Angluin and Laird [1] coined the terminology PAC learning. A more precise definition follows.

Let X be the instance space of interest. The target concept f maps X into {0,1}. Similarly, for any other concept h, we have

$$h : X \rightarrow \{0,1\}.$$

The error, d(h,f), of a learned concept h is the probability of the instances incorrectly classified by h. That is,

$$d(h,f) = \text{Prob}\{x \in X : h(x) \neq f(x)\}.$$

Prob{ } is determined by an arbitrary sampling distribution, D, over X. Learning is accomplished by processing a learning procedure on a sample of instances called the training sample. Sampling is assumed to be with replacement with samples drawn independently.

For $0<\varepsilon, \delta<1$, a learning procedure is said to be a probably approximately correct (with respect to D) identification of the target concept f if

$$\text{Prob}\{d(h,f)\geq\varepsilon\} \leq \delta.$$

We say that above learning procedure is an efficient PAC identifier if it is a polynomially bounded algorithm which identifies a concept from a random sample, whose size is polynomially bounded.

As we see in the definition, the PAC learning model is defined for {0,1}-valued functions. Haussler [11] gives a generalization of the PAC learning model that is based on statistical decision theory and can be applied for multi-valued discrete functions, real-valued functions and vector-valued functions. In this generalized mode the learning system receives randomly drawn examples, each example consisting of an instance $x\in X$ and an outcome $y\in Y$. The learning system finds a hypothesis

$$h : X \rightarrow A$$

that specifies the appropriate action $a\in A$ to take for each instance x, in order to minimize the expectation of a loss function L(y,a). Here X, Y and A are arbitrary sets, L is a real-valued function, and examples are generated according to arbitrary joint distribution on $X\times Y$.

3.4 The Performance of a Learning Algorithm

As we have seen in the definition of PAC learning, two measures of learning performance are relevant. The first is sample complexity, and the second is computational complexity.

1) Sample Complexity : The sample complexity is the number of random examples needed to produce a hypothesis that with high probability has small error. It is defined by taking the number of random examples needed in the worst case over all the target concepts in the class and all the probability distributions on the instance space.

2) Computational Complexity : The computational complexity is the worst case computation time to produce an hypothesis from a sample of a given size.

We use big-O notation to denote both complexities.

Vapnik [44] was the first to give a characterization of the sample complexity of a learning

alogrithm. Below Theorem 3.2 gives a sufficient sample size for PAC identification. (Also see [3])

Theorem 3.2 : Let N be the number of rules in the hypothesis space H. Let f be the target concept. If h is any hypothesis that agrees with at least

$$m = (1/\varepsilon) \ln (N/\delta)$$

random samples, then

$$\text{Prob}\{d(h,f) \geq \varepsilon\} \leq \delta.$$

However, the above bound is very loose, and if the size of an hypothesis space is not finite, such as the set of intervals on the real line, the method cannot be applied.

So, there is a need to improve this bound. To improve the sample complexity we may use several different measures of a hypothesis space other than the number of rules in the hypothesis space [9, 44]. Two other combinatorial parameters which measure the characteristics of a hypothesis space are given below.

Definition 3. 3 : (Growth function, VC dimension)

1. Growth function ($\pi_H(m)$) : The growth function, $\pi_H(m)$, is the maximum number of dichotomies (i. e., the maximum number of ways of partitioning a set into a set of positive instances and a set of negative instances) induced by hypotheses in H on any set of m instances.

2. Vapnik-Chervonenkis dimension of H (VCdim(H)) : Let I be a set of instances in X. If H induces all possible $2^{|I|}$ dichotomies of I, then we say that H shatters I. The Vapnik-Chervonenkis dimension of H, denoted by VCdim(H), is the cardinality of the largest finite subset I of X that is shattered by H, or equivalently, the largest m such that the Growth function $\pi_H(m) = 2^m$. If arbitrarily large subsets of X can be shattered, then $VCdim(H) = \infty$.

The following theorem gives one of the main results using the VC dimension. (See [4, 9] for more details)

Theorem 3. 4 : [4] Let H be any nonempty hypothesis space, and let d be the VC dimension of H, where d is finite. Let f be the target concept. For any $0 < \varepsilon < 1$, if h is any hypothesis that agrees with at least

$m = \min [\ (1/\varepsilon)[\ln(1/\delta) + \ln |H|, (1/\varepsilon)\{4 \log(2/\delta) + 8d \log(13/\varepsilon)\}]$ random samples, then

$$\text{Prob}\{d(h,f) \geq \varepsilon\} \leq \delta.$$

Below we give some examples of the above three measures for the hypothesis space, and give an illustration of Theorem 3.2 and Theorem 3.4.

Example 1 : the case of $|H| < \infty$

Consider the following attributes of a firm. Suppose we are to characterize successful firms using the following list of attributes.

Attributes	Values
INDUSTRY	TYPE ELECTRONICS
	BANKING
	AUTOMOBILE
SIZE	LARGE
	MEDIUM
	SMALL
STRATEGIC PLANNING DEPARTMENT	YES
	NO
MIS DEPARTMENT	YES
	NO
CURRENT RATIO	GREATER THAN 3.0
	BETWEEN 1.5 AND 3.0
	LESS THAN 1.5
DEBT-EQUITY RATIO	GREATER THAN 0.7
	LESS THAN OR EQUAL TO 0.7

Suppose H is a conjunctive concept. Since each attribute can either be a term of a conjunctive concept or not, the number of rules in H is $|H| = 3^3 4^3 = 1{,}728$. That is, $N = 1{,}728$.

Hence, by Theorem 3.2, the learned concept h has error ε with probability $1-\delta$ after

$(1/\varepsilon)\ (\ln(1/\delta) + \ln 1{,}728)$

$= (1/\varepsilon)\ (\ln(1/\delta) + 7.455)$

random independent examples, regardless of the underlying distribution governing the generation of these examples. Note that the number of examples required grows slowly compared to the size of the hypothesis space.

For $\varepsilon = 0.1$ and $\delta = 0.05$, Theorem 3.2 shows that the number of examples required for PAC learning is

$$m = (1/\varepsilon)(\ln(1/\delta) + \ln 1{,}728) = 196.$$

In other words, a learned concept h has error less than 10% with probability 95% after 196 random examples, regardless of the underlying distribution.

From Haussler [9] Theorem 3.6, the VCdim(H) for pure conjunctive hypothesis satisfies

$$n \leq VCdim(H) \leq 2n$$

where n is the number of attributes. Thus

$$6 \leq VCdim(H) \leq 12$$

For $\varepsilon = 0.1$ and $\delta = 0.05$, Theorem 3.4 shows that the number of examples required for PAC learning is

$$\begin{aligned} m &= \min [\ (1/\varepsilon)(\ln(1/\delta)+\ln 1{,}728),\ (1/\varepsilon)(4\log(2/\delta) + 8d\log(13/\varepsilon))] \\ &= \min [\ (1/0.1)(\ln(1/0.05)+\ln 1{,}728), \\ &\quad (1/0.1)(4\log(2/0.05) + 8*12\log(13/0.1))] \\ &= 196 \end{aligned}$$

In this case, we could not improve the sample complexity because VCdim(H) is not less than $\ln|H|$.

Example 2 : the case of $|H| = \infty$

Suppose we wish to learn the range of liabilities to assets ratios within which successful companies operate. The instance space X is the interval [0,1]. Let the hypothesis space H be the intervals [x, y] with $0 \leq x \leq y \leq 1$ plus the empty set. Since there are an infinite number of intervals in [0, 1], $|H| = \infty$. The growth function for H is determined as follows.

Consider the single example with value 0.6. The instance $0.60 \in X$ can be labelled as + (a positive example) by the concept [0.5, 1] and − (a negative example) by the concept [0, 0.5]. Hence $\pi_H(1) = 2 = 2^1$.

Consider two examples with values 0.3 and 0.5. The following four concepts give all different partitionings of two examples.

[0.1, 0.4] gives (+,−) :
[0.4, 0.7] gives (−,+) ;
[0.2, 0.7] gives (+,+) ; and
[0.4, 0.5] gives (−,−).

Thus $\pi_H(2) = 4 = 2^2$.

Now consider three examples with values a, b, and c, where a<b<c. Since no disjunction of intervals is allowed, no concept in H can classify (a, b, c) as (+,−,+), but all other classification combinations are possible.

Thus $\pi_H(3) = 7 < 2^3$.

Therefore, VCdim(H) = 2.

Since there are infinite number of intervals on the real line, the number of rules in H is |H| = ∞. That is, N = ∞.

Hence, by Theorem 3.2, the learned concept h has error ε with probability $1-\delta$ after ∞ random independent examples.

For ε = 0.1 and δ = 0.01, Theorem 3.4 shows that the number of examples required for PAC learning is

$$m = \min [\infty, (1/0.1)\{4 \log(2/0.01) + 8*2 \log(13/0.1)\}]$$
$$= 1429.3$$

In this case, we have improved the sample complexity because VCdim(H) is considerably less than ln|H|.

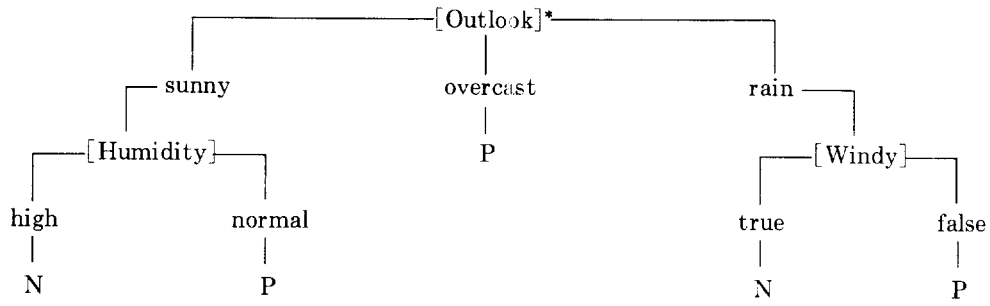
4. A theoretical analysis of Decision Tree Induction Methods

A decision tree can be expressed as a disjunctive concept. Each path in a decision tree corresponds to a conjunction of variables (with values), and all pathes having the same class in their leaves can be combined with disjunctions. For example,

{((Outlook=sunny) and (Humidity=normal)) or
(Outlook=overcast) or ((Outlook=rain) and Windy=false))}

is a disjunctive concept, P, represented by a decision tree in Figure 4.1. Here we have four attributes having values in parentheses.

1. Outlook (sunny, overcast, rain)
2. Temperature (hot, mild, cool)
3. Humidity (high, normal)
4. Windy (true, false)



* : [ ] denotes an attribute.

[Figure 1] A decision tree learned from a training set

4.1 Learning Disjunctive Concepts

We begin with the definition of the DNF (Disjunctive Normal Form) concept.

Definition 4.0 : A disjunctive normal form (DNF) expression is any sum $m_1+m_2+\cdots+m_r$ of monomials where each monomial m_i is a product of literals. Here "sum" means an inclusive OR operation and "product" means an AND operation. A literal is either a variable (i. e., an attribute with value) or the negation of a variable. An expression is monotone if no variable is negated in it.

For the case of monotone DNF expressions, having a bound, k, on the length of each disjunct (we call this a k-DNF concept), Valiant [42, 43] showed that a PAC concept can be learned in polynomial time from negative examples with the sample complexity $O(n_k)$, where n is the number of variables. Haussler [9] improves this result by using a dual greedy method which is a variant of the "star" methodology of Michalski [21]. The improved bound is $O((\log kn)^2)$.

Rivest [34] showed k-DNF concepts without a monotonicity restriction are polynomially learnable using decision lists. A decision list is an extended "if-then-elseif-else" rule, where the tests in "if" parts are conjunctions of literals drawn from 2n literals. (See Rivest 1987 for the precise definition.) Compared to decision trees, decision lists have a simpler structure, but the complexity of the decisions allowed at a node is greater. Let k-DL be the set of all Boolean functions defined by decision lists, where each function in the list is a term of size at most k. k-DNF(n) is a proper subset of k-DL(n), where n is the number of variables used in the expression. Rivest shows that k-DL is polynomial-sized and polynomially-identifiable. If a class of formulae is polynomial-sized and polynomially identifiable, then it is polynomially learnable. How-

ever, computation of the "shortest" decision list consistent with a given sample is an NP-hard problem [34].

4.2 Learning Decision Trees

A decision tree has much expressive power in the sense that it is concise and that there is no limitation on the attributes and classifications allowed, and is a more complex structure than the DNF concepts or decision lists. Therefore little theoretical research has been done on decision trees, and most research on learning a decision tree is based on heuristic reasoning.

There are a number of algorithms for learning decision trees [23, 24, 28, 30, 31, 32, 33, 41]. Most algorithms involve three main stages:

1) Construct a complete tree able to exactly classify all the examples.
2) Prune this tree to give statistical reliability.
3) Process the pruned tree to improve understandability.

Some algorithms adopt pruning techniques while they construct a decision tree.

In the following we give a theoretical analysis on decision tree induction algorithms. One of the popular learning algorithm is ID3. ID3 is easy to implement and has given excellent results in a number of applications. ID3 operates on domains where instances can be represented by a finite vector of attributes.

ID3 construct a decision tree as follows:

Algorithm 4.0 : (ID3)

1. If all instances are either positive or negative examples, then the tree is a single leaf node of corresponding sign.
2. Otherwise
   a. Let i be an attribute that optimizes some criteria for choosing an attribute. Create a node labelled with i.
   b. Partition the instances into k groups. For each partition, form a branch from the node to a decision tree recursively constructed.

In Step 2.a a large number of different criteria have been used, including information measure [32], a chi-square contingency table statistic [22], probability measures [23], gain-ratio [32], Marshall correlation [19], and others [23].

In Step 2.b., partitioning can be accomplished in many different ways. For nominal or discrete

attributes, the most common partitioning is along each possible value of attributes. For attributes with a large number of values including real valued attributes, the instances are typically split into two.

To determine the sample complexity for PAC-learning using Theorem 3.4, we have to determine the Vapnik-Chervonenkis dimension of the hypothesis space H.

Let C_i be the cardinality of the set of the permissible values for attribute i, $1 \le i \le n$, where n is the number of attributes that define the domain X.

Then,

$$d = C_1 \times C_2 \cdots \times C_i \times \cdots \times C_n$$

is the total number of distinct instances in X.

Since each distinct instance can be classified as positive or negative, the total number of concepts that can be defined by decision trees on X is

$$2^d = |H|.$$

For a sample that consists of d distinct instances (i. e., the whole instance space X), the maximum number of possible concepts induced by H is 2^d, therefore

$$VCdim(H) = d.$$

Applying Theorem 3.2 and 3.4 with $|H|=2^d$, and $VCdim(H)=d$, we can derive the sample size needed for probably approximately correct learning using inductive decision trees. Following Theorem 4.1 summarizes this.

Theorem 4.1 : (Sample complexity for ID3)

For any given ε and δ, with $0 \le \varepsilon, \delta \le 1$, if the sample size is at least

$$[\ \ln(1/\delta) + d \ln 2\ ]\ /\varepsilon,$$

then an induced decision tree using ID3 will have error less than or equal to ε with probability greater than $1-\delta$.

As we see from the above Theorem 4.1, ID3 does not satisfy efficiency property of PAC learning since sample complexity is not polynomial on the number of attributes, n.

In the following we will discuss PAC learning algorithms for decision tree induction. First PAC learning algorithm for decision tree induction was introduced by Ehrenfeucht and Haussler [8]. We need the following definition of the rank of a decision tree to show a theoretical result.

Definition 4.2 : (the rank of a binary decision tree)

The rank of a binary decision tree Q, denoted r(Q), is defined as follows :

(i) If Q=0 or Q=1 then r(Q)=0.

(ii) Else if r_0 is the rank of the 0-subtree of Q and r_1 is the rank of the 1-subtree, then

$$r(Q) = \begin{cases} \max(r_0, r_1) & \text{if } r_0 \neq r_1 \\ r_0 + 1 \ (= r_1 + 1) & \text{otherwise} \end{cases}$$

Let T_n^r be the set of all binary decision trees over the attribute set V_n of rank at most r, and let F_n^r be the set of Boolean functions on the instance space X_n that are represented by trees in T_n^r.

Theorem 4.3 provide sample complexity of a decision tree induction algorithm, Find(S,r). (See [8] for the details). Here, S and r denote sample, and the rank of a decision tree, respectively.

Theorem 4.3 : For any $n \geq r > 0$, any target function $f \in F_n^p$, $p \leq n$, an equally likely probability distribution D on X_n and any $0 < \varepsilon, \delta < 1$, given a sample S derived from a sequence of

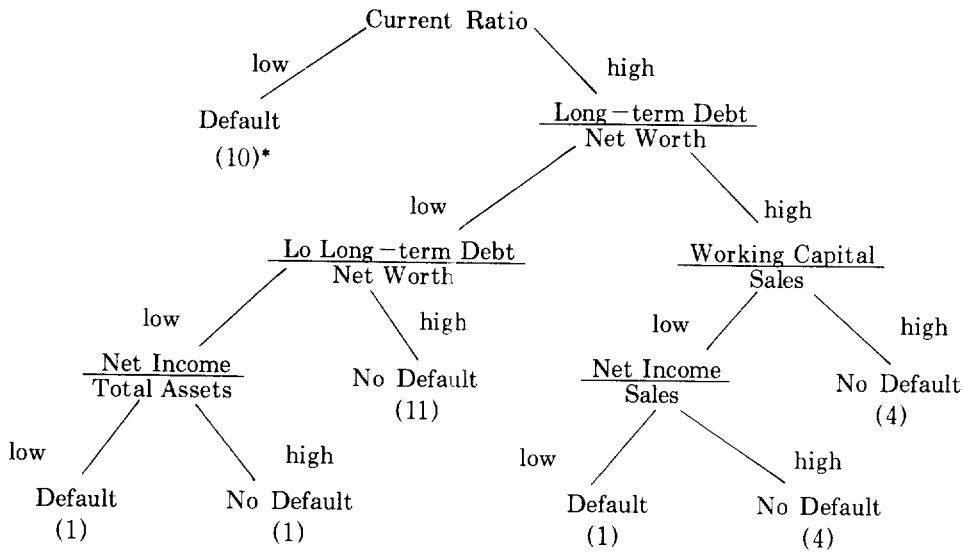
$$m \geq (1/\varepsilon)\{(en/r)^r \ln(8n) + \ln(1/\delta)\}$$

random examples of f chosen independently according to D, with probability $1-\delta$, Find(S,r) produces a hypothesis $h \in F_n^r$ that has error at most ε.

As we see from the above sample complexity, the above learning algorithm satisfies efficiency property of PAC learning paradigm.

In the following we give an example of a decision tree to illustrate the above result. Here we use a decision tree predicting loan default of companies, from Messier and Hansen [20]. In Figure 4.2 we have altered their final tree to express all the variables as binary variables. The exact meaning of high and low for each attribute is given below :

Attributes	low	high
Current Ratio	< 1.912	≥ 1.912
Long-term Debt/Net Worth	< .486	≥ .486
Lo Long-term Debt/Net Worth	< .046	≥ .046 and < .486
Working Capital/Sales	< .222	≥ .222
Net Income/Total Assets	< .100	≥ .100
Net Income/Sales	< .010	≥ .010



* : The number of examples in each terminal node.

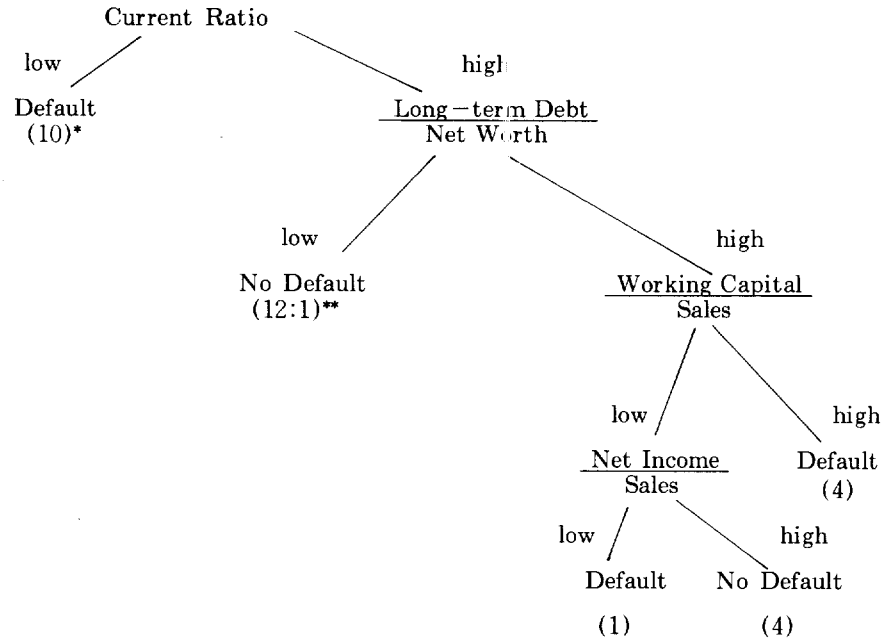
[Figure 4.2] Decision tree predicting loan default

Now we give the sample size sufficient for PAC learning of the above decision tree. Here the number of attribute is 6, and the rank of the decision tree is 2. Therefore, by Theorem 4.3, the number of examples, m, sufficient for learning a decision tree of rank two is

for $= 0.5$ and $\delta = 0.01$, m $= 525$, and
for $= 0.1$ and $\delta = 0.01$, m $= 2{,}621$.

The above algorithm assumes that there is a decision tree which classifies all the examples correctly. However, the assumption of perfect classification does not hold in business environment in general since business data are often incomplete, and exposed to noise. Also, there could be some hidden attributes which affect decision making considerably. To overcome this problem, pruning techniques are widely used in decision tree induction methods. There are a number of pruning techniques [24, 28, 33]. In the following we give a theoretical analysis considering pruning.

Suppose we need a more concise decision tree of rank one from the above decision tree of rank two in Figure 4.2. By a PAC pruning algorithm Prune(r,k,Q,S), from Kim [13], following decision tree is obtained. Here S denotes sample, Q denotes decision tree, k and r are the rank of desired decision tree, and the rank of original decision tree, respectively.



* : The number of examples in each terminal node.

** : 12 examples are in No Default, One example is in Default.

[Figure 4.3] A pruned decision tree predicting loan default

Following Theorem 4.4 and 4.5 provide sample complexity of a decision tree induction algorithm with pruning. The details are found in Kim [13].

Theorem 4.4 : For any $n \geq r > 0$, any target function $f \in F_n^p$, $p \leq n$, an equally likely probability distribution D on X_n and any $0 < \varepsilon, \delta < 1$, given a sample S derived from a sequence of

$$m \geq [2/\{\varepsilon^2 (1-2\mu_{n,r})^2\}] \{(en/r)^r \ln(8n) + \ln(2/\delta)\}$$

random examples of f chosen independently according to D, with probability $1-\delta$, Find(S,r) and Prune(r,k,Q,S) using tree labelling produces a hypothesis $h \in F_n^r$ that has error at most ε. Here e is the base of the natural logarithm, and

$$\mu_{n,r} = 0.5 - (0.5)^{n-r+1} \text{ for } n \geq r = 1$$
$$\mu_{n,r} = 0.5 - \{1+(n-r)(0.5)^2\}(0.5)^{n-r+1} \text{ for } n \geq r > 1.$$

Theorem 4.4 shows that the decision tree with rank at most r on n variables can be learned with pruning with accuracy $1-\varepsilon$ and confidence $1-\delta$ in time polynomial in $1/\varepsilon$, $1/\delta$, $1/(1-2\mu_{n,r})$ and n for fixed rank r, allowing one unit of time to draw each random examples. Thus, pruning has increased the number of examples we must obtain, but still retains its polynomial sampling characteristic. The above sample sizes in Theorem 4.4 can be reduced by using Laird's [18] improved bound on learning from noisy examples. Theorem 4.5 presents this bound.

Theorem 4.5 : Assume $0<\varepsilon, \delta<1/2$. For any $n\geq 1>0$, any target function $f\in F_n^r$, $p\leq n$, an equally likely probability distribution D on X_n and any $0<\varepsilon, \delta<1/2$, given a sample S derived from a sequence of

$$m \geq \lceil 1/\{\varepsilon(1-\exp(-(0.5)(1-2\mu_{n,r})^2))\}\rceil \{(en/r)^r\ln(8n)+\ln(1/\delta)\}$$

random examples of f chosen independently according to D, with probability $1-\delta$, Find(S,r) and Prune(r,k,Q,S) using tree labelling produces a hypothesis $h\in F_n^r$ that has error at most ε.

Now we give sample sizes sufficient for learning a decision tree with pruning. We use the above decision tree of rank one pruned from an induced decision tree of rank two. Here n=6, k=2, and r=1. Then, by Theorem 4. 4, the number of examples, m, sufficient for learning a decision tree of rank one is

for $\varepsilon = 0.5$ and $\delta = 0.01$, m = 560,609, and
for $\varepsilon = 0.1$ and $\delta = 0.01$, m = 14,015,240.

The above sample sizes are very large because of the loose bounds of $\mu_{n,r}$ and F_n^r. Since k=2, $\mu_{6,1}$ can be substituted by the least upper bound of the pruning error, $\eta_{2,1}=0.25$. In fact, $|F_n^r|\leq N=2^{64}$ for any value of r. With these tighter bounds the number of examples, m, sufficient for learning reduces to

for $\varepsilon = 0.5$ and $\delta = 0.01$, m = 6,356 and
for $\varepsilon = 0.1$ and $\delta = 0.01$, m = 158,895.

By using Theorem 4.5 and tighter bounds for $\mu_{n,r}$ and F_n^r, the sufficient sample size could be reduced as follows :

For $\varepsilon = 0.499$ and $\delta = 0.01$, m = 833,
for $\varepsilon = 0.2$ and $\delta = 0.01$, m = 2,084, and

for $\varepsilon = 0.1$ and $\delta = 0.01$, m=4,168 are obtained as the number of examples suffcient for learning.

Hence, the learned concept(that is, a pruned decision tree of rank at most one) h has error less than 10% with probability greater than 99% after 4,168 random independent examples.

5. Discussions

Business environments are dynamically changing, there are many variables to be considered, some hidden attributes may affect business decisions seriously. Also, business data contains noise in the attributes or noise in the classifications.

PAC learning theory provides a theoretical framework to analyze AI induction algorithms including decision tree induction algorithms. We are able to analyze the concept accuracy (prediction accuracy), computational efficiency (learning or training time) and sample complexity of various induction algorithms by this learning framework.

There are other important researches for the usage of AI induction algorithms for business environments. As we see in the above examples, sample size sufficient for PAC learning is often very large, while examples and data are often very costly in many business environments. In practice, many AI induction algorithms applied to business decisions (mainly on classification tasks) have used very small number of examples [6, 20]. For those cases, posterior error analysis methods are developed and used to assess the concept accuracy of AI induction algorithms [14, 16].

Also, there are researches to find theoretical conditions of pruning where pruning is beneficial for concept accuracy as well as concept simplification [15].

With the introduction of learning theory to business domains, rules or concepts learned from induction algorithms are able to be used in business decision making or in expert systems with more rigorous error estimates and predetermined confidence level.

References

[1] Angluin, D. and Laird, P., "Learning From Noisy Examples," *Machine Learnng*, Vol. 2, (1988), pp. 343-370

[2] Angluin, D. and Smith, C. H., "Inductive Inference: Theory and Methods," *Computing Surveys*, Vol. 15 (1983), pp. 237-269.

[3] Blumer, A., Ehrenfeucht, A., Haussler, D. and Warmuth, M., "Occam's Razor," *Information Processing Letters*, Vol. 24 (1987), pp. 377-380.

[4] Blumer, A., Ehrenfeucht, A., Haussler, D. and Warmuth, M., "Learnability and the Vapnik-Chervonenkis Dimension," Technical Report UCSC-CRL-87-20, University of California, Santa Cruz, CA. 1987.

[5] Braun, H. and Chandler, J. S., "Predicting Stock Market Behavior Through Rule Induction : An Application of the Learning-from-Example Approach," *Decision Sciences*, Vol. 18 (1987), pp. 415-429.

[6] Cohen, P. R. and Feigenbaum, E. A., *The Handbook of Artificial Intelligence, Vol* Ⅲ, Reading, MA : Addison-Wesley, 1982.

[7] Davis, R., "TEIRESIAS: Applications of Meta-Knowledge," *in Knowledge Based Systems in Artificial Intelligence* (pp. 227-408), R. Davis and D. Renat (Eds.), New York : McGraw-Hill, 1982.

[8] Ehrenfeucht, A. and Haussler, D., "Learning Decision Trees From Random Examples," *Proceedings of the 1988 Workshop on Computational Learning Theory* (pp. 182-194), San Mateo, CA: Morgan Kaufmann, 1988.

[9] Haussler, D., "Quantifying Inductive Bias AI Learning Algorithms and Valiant's Learning Framework," *Artificial Intelligence*, Vol. 36 (1988), pp. 177-221.

[10] Haussler D., "Learning Conjunctive Concepts in Structural Domains," *Machine Learning*, Vol. 4 (1989), pp. 7-40

[11] Haussler, D., "Decision Theoretic Generalization of the PAC Model for Neural Net and Other Learning Applications," Technical Report, UCSC-CRL-91-02, University of California, Santa Cruz, 1990.

[12] Johnson, D. E., "What Kind of Expert Should a System Be?," *Journal of Medicine and Philosophy*, Vol. 8 (1983), pp. 77-97.

[13] Kim, H., *PAC-learning a Decision Tree with Pruning*, Ph. D. Dissertation, Department of Decision and Information Scineces, University of Florida, 1992.

[14] Kim, H. and Koehler, G., "The Accuracy of Decision Tree Induction in a Noisy Domain for Expert Systems Construction", *Proceedings of '93 Korea/Japan Joint Conference on Expert Systems*, pp. 303-313, Seoul, Korea, 1993.

[15] Kim, H. and Koehler, G., "An Investigation on the Conditions of Pruning an Induced Decision Tree", *The European Journal of Operational Research*, forthcoming.

[16] Kim, H. and Koehler, G., "A Theoretical Analysis on the Pruning Techniques for Decision

Tree Induction", *Proceedings of the Decision Science Institute Second International Conference*, pp. 709-712, Seoul, Korea, June 1993

[17] Koehler, G. J. and Majthay, A., "Generalization of Quinlan's Induction Method," Department of Decision and Information Sciences, University of Florida, Unpublished Manuscript, 1988.

[18] Laird, P., *Learning From Good Data and Bad*. Doctoral Dissertation, Department of Computer Science, Yale University, New Haven, CT, 1987.

[19] Marshall, R., "Partitioning Methods for Classification and Decision Making in Medicine," *Statistics in Medicine*, Vol. 5 (1986), pp. 517-526.

[20] Messier, W. F. and Hansen, J. V., "Inducing Rules for Expert Systems Development," *Management Science*, Vol. 34, No. 12 (1988), pp. 1403-1415.

[21] Michalski, R. S., "A Theory and Methodology of Inductive Learning," *Artificial Intelligence*, Vol. 20 (1983), pp. 111-161.

[22] Mingers, J., "Expert Systems--Experiments with Rule Induction," *Journal of the Operational Research Society*, Vol. 37 (1986), pp. 1031-1037.

[23] Mingers, J., "An Empirical Comparison of Selection Measures for Decision Tree Induction," *Machine Learning*, Vol. 3 (1989), pp. 319-342.

[24] Mingers, J., "An Empirical Comparison of Pruning Methods for Decision Tree Induction," *Machine Learning*, Vol. 4 (1989), pp. 227-243.

[25] Mitchell, T. M., "Generalization as Search," *Artificial Intelligence*, Vol. 18 (1982), pp. 203-226.

[26] Musen, M. A., "Automated Support for Building and Extending Expert Models," *Machine Learning*, Vol. 4 (1989), pp. 347-375.

[27] Natarajan, B. K., *Machine Learning : A Theoretical Approach*, San Mateo, CA : Morgan Kaufmann, 1991.

[28] Niblett, T., "Constructing Decision Trees in Noisy Domains," *Proceedings of the Second European Working Session on Learning* (pp. 67-78), Bled, Yugoslavia : Sigma Press, 1987.

[29] Nunez, M., "The Use of Background Knowledge in Decision Tree Induction," *Machine Learning*, Vol. 6 (1991), pp. 231-250.

[30] Quinlan, R., "Discovering Rules from Large Collection of Examples : A Case Study," In D. Michie(Ed.), *Expert Systems in the Microelectronic Age* (pp. 168-201). Edinburgh: Edinburgh University Press, 1979.

[31] Quinlan, R., "The Effect of Noise in Concept Learning," In R. S. Michalski, J. Carbonell, T. Mitchell(Eds.), *Machine Learning : An Artificial Intelligence Approach*. Vol. II, Chap-

ter 6, Los Altos, CA: Morgan Kaufmann, 1983.

[32] Quinlan, R., "Induction of Decision Trees," *Machine Learning*, Vol. 1 (1986), pp. 86-106.

[33] Quinlan, R., "Simplifying Decision Trees," *International Journal of Man-Machine Studies*, Vol. 27 (1987), pp. 221-234.

[34] Rivest, R., "Learning Decision Lists," *Machine Learning*, Vol. 2, No. 3 (1987), pp. 229-246.

[35] Shaw, M. J. and Gentry, J. A., "Using an Expert System with Inductive Learning to Evaluate Business Loans," *Financial Management*, Vol. 17(1988), pp. 45-56.

[36] Shaw, M. J. and Gentry, J. A., "Inductive Learning for Risk Classification," *IEEE Expert*, Vol. 5(1990), pp. 47-53.

[37] Shaw, M. J., Gentry, J. A. and Piramuthu, S., "Inductive Learning Methods for Knowledge-Based Decision Support : A Comparative Analysis," *Computer Science in Economics and Management*, Vol. 3 (1990), pp. 147-165.

[38] Simon, H., "Why Should Machines Learn?" In R. S. Michalski, J. Carbonell, T. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*, Vol. I (pp. 25-37), Palo Alto, CA: Tioga, 1983.

[39] Spangler, S., Fayyad, U. M. and Uthurusamy, R., "Induction of Decision Trees from Inconclusive Data," *Proceedings of the 6th International Conference on Machine Learning* (pp. 146-150), San Mateo, CA: Morgan Kaufmann, 1989.

[40] Tsai, L. and Koehler, G. J., "The Accuracy of Concepts Learned from Induction," *Decision Support System*, forthcoming.

[41] Utgoff, P., "Incremental Induction of Decision Trees," *Machine Learning*, Vol. 4 (1989), pp. 161-186.

[42] Valiant, L. G., "A Theory of the Learnable," *Communications of the ACM*, Vol. 27, No. 11 (1984), pp. 1134-1142.

[43] Valiant, L. G., "Learning Disjunctions of Conjunctions," *Proceedings of the 9th International Joint Conference on Artificial Intelligence* (Vol. 1, pp. 560-566), Los Angeles, CA: Morgan Kaufmann, 1985.

[44] Vapnik, V. N., *Estimation of Dependencies Based on Empirical Data*, New York: Springer-Verlag, 1982.

[45] Weiss, S. M. and Kulikowski, C. A., *Computer Systems that Learn*, Los Altos, CA: Morgan Kaufmann, 1991.