Towards A Practical Estimate Of Training Sample Size

by

Steve G. Romaniuk

Department of Information Systems & Computer Science

National University of Singapore

10 Kent Ridge Crescent

Singapore 0511

e-mail: stever@iscs.nus.sg

Abstract

The purpose of this paper is to introduce a simple learning model that allows one to draw conclusions about the number of *distinct* training examples required to learn some boolean function with at least accuracy α and probability δ across a general class of learning algorithms. The motivation for this work stems from the inability of learning theoretical models to suggest reasonable sample bounds. Reducing sample size is essential in the wake of expected costs for labeling patterns by a teacher (e.g. human expert). The derived results are then extended from learning functions to learning concepts to make the analysis more realistic. The importance of domain specific knowledge in learning concepts is discussed and incorporated into the model in form of identifying *impossible* training patterns. Several possible sources for these *impossibilities* are pointed out. The paper is concluded with 2 representative examples.

1 Introduction

Since a paper published by Valiant in 1984 [20] numerous work has been conducted to more formally study the process of learning using a sound mathematical model [8, 9, 12, 2, 3, 4, 7, 10, 11, 13]. Valiant's work resulted in Probably Approximately Correct (PAC) learning theory. In this model it is assumed that there exists a distribution $\mathcal{P}$ from which a hypothetical teacher both randomly as well as independently draws examples (with replacement) and classifies each of these as either being positive or negative examples of some concept that is to be learned by algorithm L. The purpose of a learning algorithm is to recognize with some probability (confidence) δ the underlying concept - according to distribution $\mathcal{P}$ - within error ϵ. More formally this is stated as:

Definition 1 *Given a feature space X and a probability distribution D defined on X. Let n be the number of binary input features and F the class of boolean functions defined on $\{0,1\}^n$. The goal is then to learn (approximate) some target function $f \in F$ using approximating function f'. A class F of boolean functions is PAC learnable from examples, if there exists a learning algorithm L such that $\forall \epsilon > 0 \wedge \delta < 1 \wedge f \in F : Prob_{x \in D}[(f(x) \neq f'(x)) > \epsilon] < \delta$. Here N, $\frac{1}{\epsilon}$, and $\frac{1}{\delta}$ are bounded by a polynomial.*

In other words, any learning algorithm L supplies with at least probability $1 - \delta$ an approximation function $f'(x)$ of $f(x)$ such that for any $x \in D$ the probability of $f'(x)$ incorrectly approximating $f(x)$ is no greater than ϵ. ϵ and δ are generally referred to as the accuracy and confidence parameters of a PAC learning algorithm [20].

It is important to note, that L need not know about $\mathcal{P}$, that is, qualitative statements concerning learning should hold for arbitrary $\mathcal{P}$. Additionally, error is only measured with respect to distribution $\mathcal{P}$. As a result of this, L is not faulted if it miss-classifies examples which have a tendency to occur rarely with respect to $\mathcal{P}$. In general, PAC theory attempts to measure the complexity of a given learning algorithm L for a particular domain. This domain can be, for example, a restricted class of concepts such as $CNF_n(k)$ (all propositional formulae over n boolean variables in conjunctive normal form with at most k literals). Apart from identifying restricted domains that are PAC-learnable, maximum bounds on the number of training patterns required are provided. These bounds are expressed as polynomials of the parameters n, $\frac{1}{\delta}$, and $\frac{1}{\epsilon}$.

1.1 Problems With PAC

A major shortcoming of PAC derived sample bounds is, they tend to have large magnitudes, which make their use impractical when applied to real world domains. This is especially true, if we associate cost-functions with obtaining a certain quantity of labeled examples required during supervised learning. In PAC the *Oracle* - supplier of labeled (classified) samples - is held ignorant. No provisions are made for the inclusion of domain specific and domain encoding knowledge. Hence, unduly large sample bounds are suggested, which need to be supplied by the oracle. Under realistic conditions, labeling of samples can not always be simply automated. [1] Quite frequently a human teacher is required and with this requirement comes a heavy penalty: time consuming labeling of training examples. It is therefore paramount to suggest feasible sample bounds.

Another limitation of many PAC derived sample bounds is that they are expressed in terms of an abstract quantity - Vapnik-Chervonenkis Dimension (d) [21, 4]. This combinatorial parameter of the class of concepts to be learned (C), needs to be determined and its value is dependent on the complexity of the concept class C. In fact in [4] it is shown that if C is a finite concept class, VC dimension of C is at most $\log |C|$. Still, if the cardinality of a concept class is large, it can adversely affect the derived bounds (Consider $|\mathcal{B}^n| = 2^{2^n}$, class of all boolean functions).

In light of these limitations, we have to seriously question the feasibility of PAC to deliver answers to the question of how many training patterns are required for learning realistic tasks. The problem of deriving suitable bounds is tied-in with PACs approach to utilize *random* examples. In all cases, examples are supplied in random order by some *oracle*, which indicates PAC analysis has to take ill-behaved distributions ($\mathcal{P}$) into account as well as good-behaved ones. This problem is further compounded by PAC's shortcoming to utilize a priori domain specific knowledge.

For numerous real-world classificatory domains additional information in the form of domain specific knowledge is obtainable. This information may be available in terms of types of training inputs, such as, possible ranges for continuous valued data, or nominal, ordinal, and hierarchical attributes. Furthermore, inputs may only occur in conjunction with certain outputs. Additional knowledge in form of rules and conditional explanations for describing similar knowledge may also be present. Finally, dealing with unknown input values, knowledge of complementary inputs, and relative importance of inputs can in many

[1] If labeling of examples were automatic, there would be no need for learning algorithms to perform this task.

cases be identified. It would seem wasteful not to incorporate all this information, if it is readily available, or can be relatively easily elicited from domain expert(s).

1.2 Outlook On New Approach

To address the problems identified for the PAC model, we propose an alternative approach to estimate sample size bounds with a given accuracy α and confidence δ. Instead of limiting ourselves to analyze restricted, abstract classes of concepts we consider the class of all boolean functions. Furthermore, we express bounds in terms of *distinct* training examples, rather than insisting on random examples. Also, we state bounds independent of the complexity of the underlying function. Once we have introduced the distinct sample model, we expand it to include concepts. In order to make the move from functions to concepts the notion of *impossible* training patterns is introduced. Next, we point out how *impossibilities* can be identified, for example, through use of domain specific and encoding knowledge.

We propose to apply the distinct sample model to real world learning problems by investigating individual domains for potential *impossibilities* and hence utilize domain specific knowledge to develop better sample bounds. Finally, two examples are provided to compare the distinct sample model with 2 PAC derived results.

2 Estimating Sample Size

In this section, we present an upper bound on the number of *distinct* training examples required to achieve at least prediction accuracy α with confidence δ independent of the type of learning algorithm utilized.

The following quantities are given:

- $\mathcal{B}^n$: class of all boolean functions of input dimension n, $|\mathcal{B}^n| = 2^{2^n}$.

- P_{Train} : set of training patterns.

- N : number of examples (input patterns) that completely describe any boolean function $f \in \mathcal{B}^n$. That is $N = 2^n$.

- $A_L(P)$: performance accuracy of L when tested on patterns included in P.

- T : number of training patterns in P_{Train}, $|P_{Train}| = T$.

- E : maximum number of errors committed by learning algorithm L after being trained on P_{Train} and tested on **all** patterns describing target function $f \in \mathcal{B}^n$.

- δ : confidence (similar to the PAC model) or probability of L obtaining at least accuracy α when tested on all patterns describing $f \in \mathcal{B}^n$.

- α : accuracy of learning algorithm L with regard to traget function f, $\alpha = \frac{N-E}{N}$.

Theorem 1 *For any learning algorithm $L \in \mathcal{L}$ and any input dimension n of boolean n-space, algorithm L trained on examples collected in P_{Train} will correctly classify* **all** *patterns describing some boolean target function $f \in \mathcal{B}^n$ with at least accuracy α and confidence $\delta < (\frac{e}{\lceil N(1-\alpha)\rceil})^{\lceil N(1-\alpha)\rceil} \frac{(N-T)^{\lceil N(1-\alpha)\rceil}}{2^{N-T}}$.*

Proof Theorem 1 *Before proceeding with the proof let us first introduce the set $F(E,T)$. $F(E,T)$ represents a collection of functions $f \in \mathcal{B}^n$ for which any given learning algorithm L will commit exactly E errors (accuracy α) after being trained on patterns contained in P_{Train} and tested on all patterns describing f. More formally we state,*

$$F(E,T) = \{f | \forall f \in \mathcal{B}^n \; |P_{Train}| = T \; \wedge \; A_L(P_{Train}) = 1 \; \wedge A_L(f) = \alpha\} \tag{1}$$

Next, we determine the cardinality of the set $F(E,T)$ where, $E \in [0, N-1] \wedge T \in [1, N]$ [2].

Since a total of N patterns is required to fully describe any boolean function f, and of these N patterns T have been determined after learning ($|P_{Train}| = T$), the outputs for a

[2]We exclude the case of $P_{Train} = \emptyset$, thus $T \in [1, N]$

total of $N-T$ patterns remain undetermined. If algorithm L is presented with any of these patterns $< I_n, \{0,1\} >$ a choice of assigning a value of 0 or 1 as the output will have to be made [3]. *Depending on the target function f_{Target}, the response is either correct or not. If a total of E errors are committed by L, then given $N-T$ undetermined patterns, there are $\binom{N-T}{E}$ possible ways these E errors could occur. In other words, $\binom{N-T}{E}$ possible functions will be recognized by L for which its predictions will contain exactly E miss-classifications. Therefore,*

$$|F(E,T)| = \binom{N-T}{E} \tag{2}$$

We further proceed by stating, confidence δ of L, delivering a performance accuracy of at **least** *α (given T), is:*

$$\delta = \frac{\sum_{i=0}^{E} |F(i, N-T)|}{2^{N-T}} \tag{3}$$

The numerator states, number of functions f for which L can obtain at least accuracy α after being exposed to exactly T training patterns. The denominator encompasses the total number of potential target functions that can be generated. For the case of $P_{Train} = \emptyset$ we assume $E = N - T = N$ and $\alpha = 0$. Finally, the ratio yields the desired confidence, or probability. Substituting Eq. 2 into Eq. 3 we obtain,

$$\delta = \frac{\sum_{i=0}^{E} \binom{N-T}{i}}{2^{N-T}} \tag{4}$$

δ can be approximated as follows,

$$\begin{aligned} \delta &< \frac{(\frac{e(N-T)}{E})^E}{2^{N-T}} \\ &= (\frac{e}{E})^E \frac{(N-T)^E}{2^{N-T}} \\ &= (\frac{e}{\lceil N(1-\alpha)\rceil})^{\lceil N(1-\alpha)\rceil} \frac{(N-T)^{\lceil N(1-\alpha)\rceil}}{2^{N-T}}. \end{aligned} \tag{5}$$

[3] An indeterminate response is automatically considered an error

Table 1: Example of $F(E,T)$ for n=3

T $\rightarrow$ E $\downarrow$	1	2	3	4	5	6	7	8
0	1	1	1	1	1	1	1	1
1	7	6	5	4	3	2	1	
2	21	15	10	6	3	1		
3	35	20	10	4	1			
4	35	15	5	1				
5	21	6	1					
6	7	1						
7	1							

q.e.d.

In Table 1 we display the cardinality of the set $F(E,T)$ for $E \in [0,7] \ \wedge \ T \in [1,8]$. For example, if T=3 there are a total of 32 possible target functions (sum of third column entries). If we allow at most E=2 errors then L can achieve at least accuracy $\alpha = \frac{3}{4}$ with confidence $\delta = \frac{1}{2}$.

3 From Functions to Concepts

From the previously listed theorem, we can make quantitative statements with regard to the number of *distinct* training examples required to learn some boolean function within accuracy α and confidence δ. Unfortunately, learning real world domains seldomly corresponds to learning functions. Not all pattern labels are correctly assigned, (indicates presence of noise) or it may even be impossible to make an assignment (outputs are not defined for certain input combinations).

The purpose of this section is to simply point out how the earlier model can be modified to make the step from learning functions to that of learning concepts. Before we continue, let us provide the following definition:

Definition 1 *A boolean concept $C_i \in \mathcal{C}^n$ of the class of all boolean concepts of input dimension n is a mapping $C_i : \{0,1\}^n \mapsto \{0,1,x\}$. That is, all input patterns $I_j \in \{0,1\}^n$ are assigned one of three possible values, $C_i(I_j) = \{0,1,x\}$, where x symbolizes an undefined value.*

Throughout the remainder of the text we refer to input patterns I_j for which $C_i(I_j) = x$ as *impossibilities*. Clearly, as the number of *impossibilities* increases, the number of potential target functions decreases. Another way of visualizing a concept is in terms of a set of functions which overlap. The overlap is determined by inputs which are defined $(C_i(I_j) = \{0,1\})$ according to the previous definition. As the number of *impossibilities* increases, so does the set of functions which describe concept C_i.

Lemma 1 expands the earlier reported result of cardinality of set $F(E,T)$ when *impossibilities* are present.

Lemma 1 *If K out of N input patterns represent* **impossibilities** *then the number of functions for which exactly E $(0 \leq E < N)$ errors are committed by any learning algorithm L after being exposed to T training patterns $(0 < T \leq N)$ is,*

$$|F(E,T)| = \binom{N-(T+K)}{E}. \tag{6}$$

Proof Lemma 1 *Every impossibility K, removes 2^K functions from the set of potential target functions that may exist (T training patterns and E errors). Now, the more target functions are possible, the higher the probability that the function realized by L is within error α equal to a* **particular** *target function. This is equivalent to saying that miss-classifying an impossibility can not result in an increase of total error (E). Consequently, we simply add all impossibilities to the total number of patterns T.*

When calculating $F(E,T)$ we consider $T' = T + K$ and determine $F(E,T')$. Since $F(E,T) = \binom{N-T}{E}$ it follows $F(E,T) = \binom{N-(T+K)}{E}$.

q.e.d.

In the presence of *impossibilities* the earlier derived bound for δ can be restated simply as,

$$\delta < (\frac{e}{\lceil N(1-\alpha)\rceil})^{\lceil N(1-\alpha)\rceil}\frac{(N-T-K)^{\lceil N(1-\alpha)\rceil}}{2^{N-T-K}} \; and \; \alpha = 1 - \frac{E}{N-K}. \tag{7}$$

Since we have established how *impossibilities* can affect the relationship between the quantities α, δ, and T we next attempt to outline potential sources.

3.1 Sources of Impossibilities

Three possible sources of *impossibilities* can be readily identified:

1. Noise
2. Domain Encoding
3. Domain Knowledge

Note, that to identify each of the 3 sources requires the use of domain specific knowledge. Noise can have a strong impact on the generation of impossibilities. For this particular case we refer to them as contradictions. Contradictions are introduced by random toggling of output bits (characterizing presence or absence of a concept) for identical input patterns. Studies involving noise analysis [17, 18, 19] have shown that this type of noise can have profound impact on the quality of the learned concept, whereas noise on the input of patterns is of a less serious nature.

Domain encoding is required to convert symbolic information into a format which can be accepted by the learning algorithm utilized. A straight forward example is the encoding of attribute/value pairs into binary format for many neural net based algorithms [14]. If an attribute can be assigned any of k possible features, a binary vector of length k is generated. The presence or absence of a feature is encoded by assigning either a 1 or a 0 to the position of the vector which corresponds to the attribute's value. By default it is understood that an attribute will only be assigned a single value at any given time. Nonetheless, by binary coding (this can also hold for other encoding schemes) the original input space is drastically increased. As a matter of fact this increase is equivalent to $2^k - k$ for each encoded attribute. In general, the increase is given by,

$$\prod_{i=1}^{n} 2^{k_i} - k_i \tag{8}$$

where, n is the number of attributes and k_i represents the number of possible values that can be assigned to the ith attribute. This artificial expansion of the original input space is due to the increase in the number of impossibilities resulting from the encoding process and is equal to amount stated in Eq. 8. Since encoding represents an artificial enlargement of the original input space, its negative impact on learning and the distinct sample model can be dampened by simply noting the cardinality of the set of all impossibilities and applying Eq. 8.

Domain knowledge encompasses various forms of information which pertain to individual tasks. Generally, the domain expert(s) provides the knowledge engineer with this type of information. In order to sufficiently reduce the required sample bound, it is paramount to have access to this knowledge. Domain knowledge includes:

- Rule Knowledge.
- Irrelevant Attribute Values.
- Irrelevant Attributes.
- Equivalences of Attribute Values.
- Mutually Exclusive Classes (Concepts).
- Physical Limitations.

4 Types Of Domain Knowledge

It is quite common that human experts can somewhat explain their decisions and provide a brief description of some of their acquired surface knowledge during an initial interview with the knowledge engineer. This knowledge can come either in the form of meta-knowledge, or particular instances of actual examples the expert recalls. It is the former that we are seeking. Identifying different forms of meta-knowledge is quintessential to recognizing additional *impossibilities* and by doing so further reduce sample complexity.

4.1 Rule Knowledge

Rule knowledge probably represents the most commonly available knowledge for classification problems. This information can come in the form of *default*, *exception* and *common knowledge* rules. *Default* rules arise whenever a set of of basic conditions need to be automatically derived. For example, when encoding moves in various board games, *default* rules are used to either avoid impossible moves or provide for a simple automatic mechanism to execute a precise sequence of moves (operations) given certain initial conditions. *Exceptional* rule knowledge surfaces whenever infrequently occurring conditions need to be addressed in large dynamic systems. This may encompass critical as well as non-critical events. Controlling a steam engine or a nuclear reactor requires rule exceptions to shutdown these devices in case of a critical condition. Finally, *common knowledge* rules encode deep knowledge, which may include textbook knowledge such as domain independent laws (e.g. physical, chemical, etc.).

Rule knowledge encodes patterns which can be utilized when determining sample size. Patterns represented by domain rules can be interpreted as additional training examples, hence the cardinality of this set of patterns can be added as an offset to the recommended cardinality of the train set.

4.2 Irrelevant Attribute Values

IAVs allow the specification of attribute values which are not defined for some context, that is, they act as place-holders. Traditionally, numerous machine learning domains are described in terms of attribute-value pairs. Individual training patterns consist of attribute vectors and their value assignments. These vectors are of fixed length. Therefore, whenever an attribute can not be specified for a concept, a default value needs to be assigned to signify that for this particular vector an attribute has no value - it is irrelevant.

4.3 Irrelevant Attributes

Irrelevant attributes are similar to IAVs. The major difference is that instead of a particular attribute value being undefined within some context, one or more attributes may be deemed irrelevant for a given concept. The problem of IAs arises from the same representational difficulties many machine learning algorithm (and pattern recognizers in general) suffer from and which were pointed out earlier for IAVs. Since a fixed dimensional vector suffices to represent training patterns, all concepts of a domain share the same basic attributes (as defined by said vector). Obviously, concepts differ in the number of defining attributes and their relative importance. Vectors, - as mentioned previously - only represent a cross-section of all attributes and in many cases a substantial number of these attributes are irrelevant.

For example, consider the waifer manufacturing process for IC chips [14]. Parametric measures are used to sample characteristics from individual waifers. Certain manufacturing faults (the concepts to be recognized) are described in terms of these parametric measurements. Faults associated with chips that utilize npn- or pnp-transistors are by their very nature mutually exclusive. Either the family of chips is based on pnp- or npn-technology. As a result of this, certain combinations of inputs (parametric measurements) can not occur in conjunction with one another, they represent *impossibilities.*

Being able to recognize irrelevant attributes allows one to derive better sample bounds for individual concepts. Apparently, learning different concepts (various degrees of difficulty) requires diverse sample size magnitudes.

4.4 Equivalence of Attribute Values

Equivalent or similar knowledge can be identified for different attribute values and is represented in form of a conditional explanation. In PROTOS [15] rules such as the following may be encoded:

if concept=apples **then** color[red] equivalent color[green];

The above rule simply states, if the concept to be learned is *apples* then the values of red and green for the attribute color are interchangeable. Consequently, if the final concept *apples* is fully described by n sample patterns containing a value of red for color then we are in fact describing $2n$ different patterns (same patterns now with color green). If we apply the preceeding equivalence rule a posteriori to learning, we eliminate half of all possible training patterns necessary to describe concept *apples*.

4.5 Mutually Exclusive Classes

Several machine learning algorithms [5, 6, 16] require output classes to be mutually exclusive. Only a single class may be associated with a given input vector. Though this may sometimes be an unrealistic assumption (real world concepts are predominantely not separable in this fashion) it is unavoidable due to algorithmic restrictions. In other cases, domains may be structured in this manner. In any event, meta-knowledge of this type can be readily exploited to reduce sample size. Consider the following: If given two concepts C_1 and C_2 we determine - utilizing the previously stated bounds on sample size - that to learn concepts C_1 and C_2 requires T_{C_1} and T_{C_2} examples, respectively.

Consulting a teacher (oracle) we obtain the T_{C_1} patterns required to effectively learn concept C_1. Instead, of asking the teacher to supply additional T_{C_2} samples to learn C_2 we can simply use the $T_{C_1^+}$ [4] patterns suggested for learning C_1. Since we know that C_1 and C_2 are mutually exclusive concepts we can simply toggle the output bits for C_1's positive patterns and use them towards learning concept C_2. If T_{C_2} is larger than $T_{C_1^+}$ we need only ask the teacher for the difference in patterns. This can of course be generalized for n mutually exclusive classes, or groups thereof.

4.6 Physical Limitations

Physical limitations result whenever certain input combinations are generated which can not be explained by the domain model itself. In general, these limitations correspond to constraints imposed on a model due to inherent restrictions in the physical world. Measuring devices and the possible range of values they can draw from, are limited due to the precision of the device, or the way it is used in conjunction with other devices.

5 Intelligent Sample Selection

To utilize the distinct sample model, it is necessary to modify the operation of the *oracle* as a supplier of training samples.

Figure 1a shows the traditional usage of the *oracle*. Learning parameters are initially specified to determine the number of random training samples required for learning some concept. A teacher then labels the patterns supplied by the *oracle* and passes them on to the appropriate learning module. If a database of training samples already exists, then the *oracle* can directly supply them to the learning module.

The new modifications introduced for the delivery of training samples to the learning system are displayed in Figure 1b. The *oracle's* task is to provide a constant stream of distinct unlabeled samples, which are passed on to a selector module. The selector module receives domain specific as well as domain encoding knowledge and employs this knowledge to prune out *impossibilities* before passing the reduced set of samples on to the teacher for

[4] The + indicates that $T_{C_1^+}$ only contains positive examples of concept C_1.

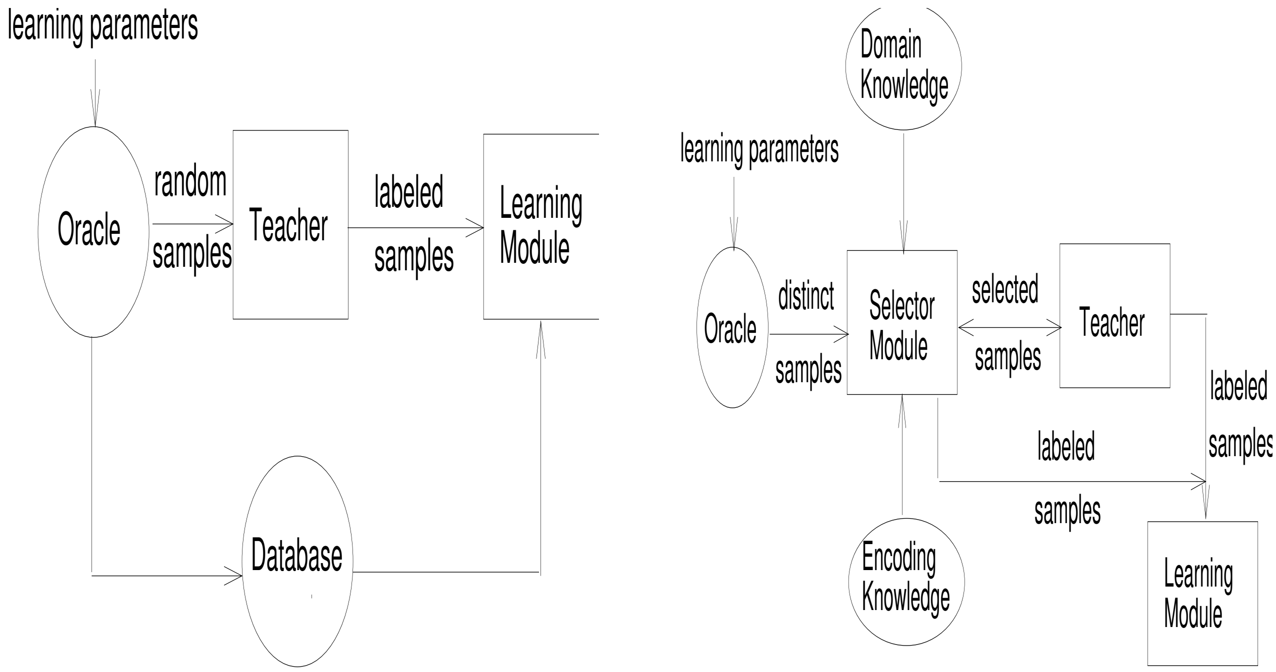

Figure 1: (a) Standard Oracle (b) Improved Oracle

labeling. The teacher may be bypassed, if the selector module can perform the labeling of samples. For example, if classes are mutually exclusive, the selector module can directly supply labels for some of the samples.

In summary, the purpose of the selector module is to prune out potential training samples which are deemed *impossibilities* by utilizing domain specific knowledge as indicated in the previous section. Pruning should significantly reduce the burden normally placed on the teacher by reducing workload. Finally, providing the learning module with fewer but sufficient patterns helps enhance learning speed.

6 Comparing Bounds: An Example

We briefly compare bounds for the distinct sample model by introducing 2 examples derived from PAC theory. The first bound is stated in [2] for a neural network algorithm that PAC learns the target functions defined by layered, 2 depth, threshold networks with n inputs connected to $0 \leq k \leq 4$ hidden threshold units. The number of examples required for

(S_{NN}) is given as,

$$m \equiv \min(n^k, k^n),\ 0 \leq k \leq 4\, and$$
$$|S_{NN}| = \max(8m\epsilon^{-1}, 8\epsilon^{-1}\log(8\delta^{-1})) \tag{9}$$

The second example is taken from [9] and describes the number of samples needed for binary decision trees (S_{DT}) of n inputs and rank r with $n \geq r \geq 1$.

$$|S_{DT}| = \frac{1}{\epsilon}\{(\frac{en}{r})^r ln(8n) + ln\frac{1}{\delta}\}, \qquad n \geq r \geq 1 \tag{10}$$

For both bounds examples in S are chosen randomly and independently. Figure 2a outlines the sample bounds for various decision tree rank values and that of the earlier derived distinct sample learning model (LM). Note, that the logarithm of the sample size is plotted and that accuracy α (ϵ for NN and DT) is shown for $\alpha \in [0.65, 1.00)$ [5]. Also, note we only present the plot of α versus $\log T$, since sample size is more sensitive with respect to α then δ.

The most striking observation we can glean from Figure 2 is the difference in growth behavior for LM, which initially increases rapidly ($\alpha < 0.65$) and then levels out, compared to DT and NN, which is reversed. Also, in case of $n = 20$ a rank of $r > 3$ results in significantly larger sample bounds for the decision tree. For the neural network bound more than 2 hidden units results in larger sample sizes (especially for higher α values).

As n is increased (Figure 3) the LM bound overtakes the $r = 3$ bound for the decision tree, but remains substantially smaller for $r > 3$. This may suggest, as n is sufficiently increased, the LM bound eventually catches up with various decision tree rank bounds. On the other hand, by increasing n we also increase the complexity of potential decision trees, that is the rank values also continue to grow. In fact, the difference between $r = \lceil \frac{n}{2} \rceil$ and the LM bound remains about the same. What this result suggests is, if n is large and we happen to know, the concept we intend to learn can be represented by a decision tree with

[5] For practical reasons we are only interested in relatively high values of α

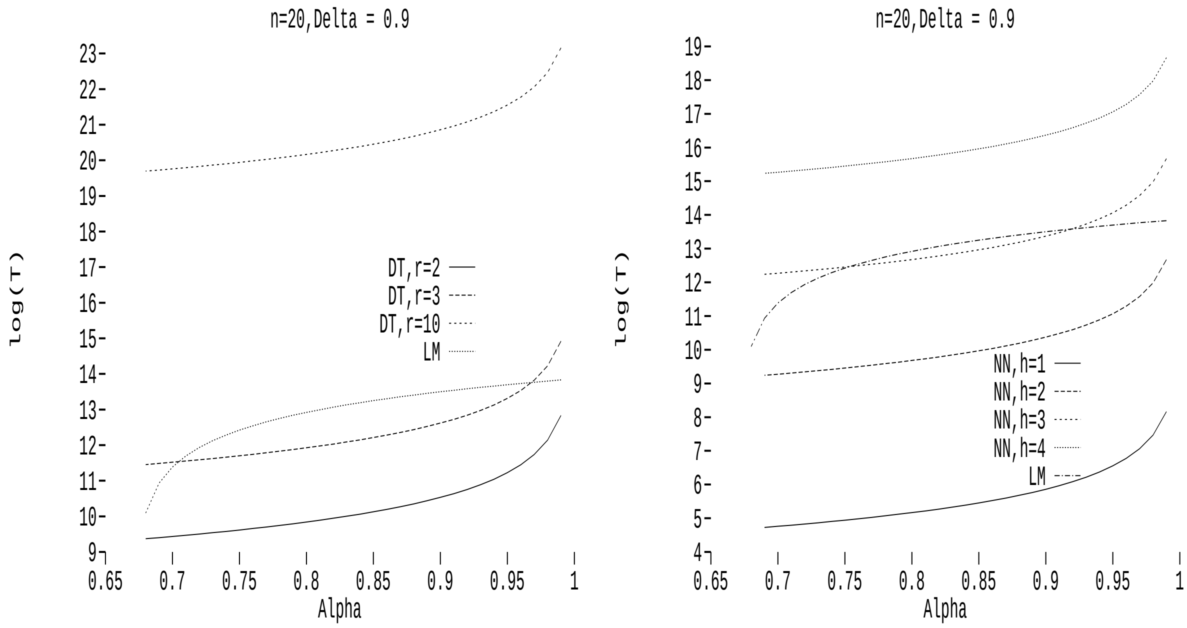

Figure 2: Accuracy α vs. $\log T$ with n=20 and $\delta = 0.9$ for LM versus (a) Decision Tree (DT) and (b) Neural Network (NN)

low rank, then the DT bounds are in general better than those suggested by LM. It seems highly unlikely, sufficient real world problems exist for which this very specific condition holds true.

To point out the importance of identifying *impossibilities* to curb sample bounds we investigate 2 tasks. The first problem is described by seven nominal attributes $A_1, A_2, \cdots, A_7$. The number of values assigned to each attribute are,

$$A_i = (5,4,4,3,3,3,3), \; i \in [1,7]. \tag{11}$$

Since we apply binary coding for representing attribute/value pairs a total of $n = 25$ bits are required to encode the 7 attributes.

Figure 4a indicates the number of patterns required using a decision tree of rank $r = 2$, a neural network with 1 hidden unit and LM with a priori knowledge (apk), that is domain encoding knowledge. Clearly, the bounds suggested by LM are significantly better than

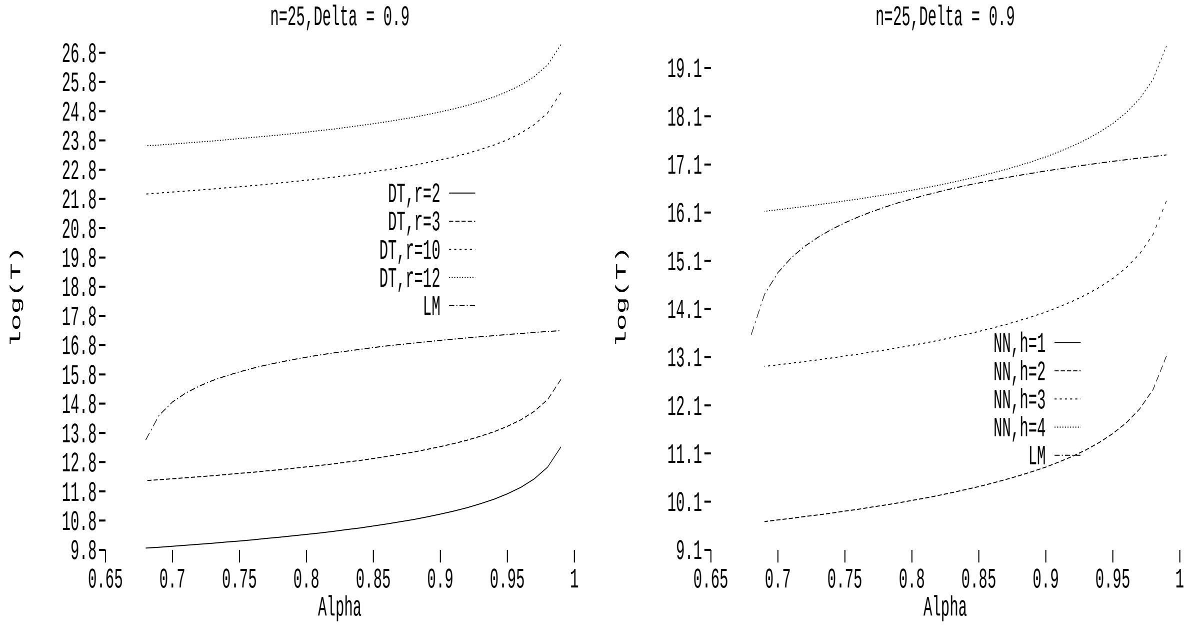

Figure 3: Accuracy α vs. $\log T$ with n=25 and $\delta = 0.9$ for LM versus (a) Decision Tree (DT) and (b) Neural Network (NN)

those proposed by the 2 PAC approaches.

For the final experiment we consider 10 attributes with 5 possible values each ($n = 50$). The results are displayed in Figure 4b. As soon as the rank of a decision tree is larger than $r > 3$ and the neural network model utilizes more than $h > 2$ hidden units the distinct sample model with a priori domain encoding knowledge provides lower sample bounds. Only very few concepts (defined by 50 binary inputs) can be represented by decision trees with a rank $r \leq 3$ or hidden units $h \leq 2$. For these few exceptions the two PAC models suggest better bounds. Finally, using domain encoding knowledge can hardly be viewed as an unfair advantage for LM, since it is readily available.

7 Summary

Deriving a learning model which allows one to make qualitative statements across a general class of learning algorithms, with respect to number of distinct training patterns required

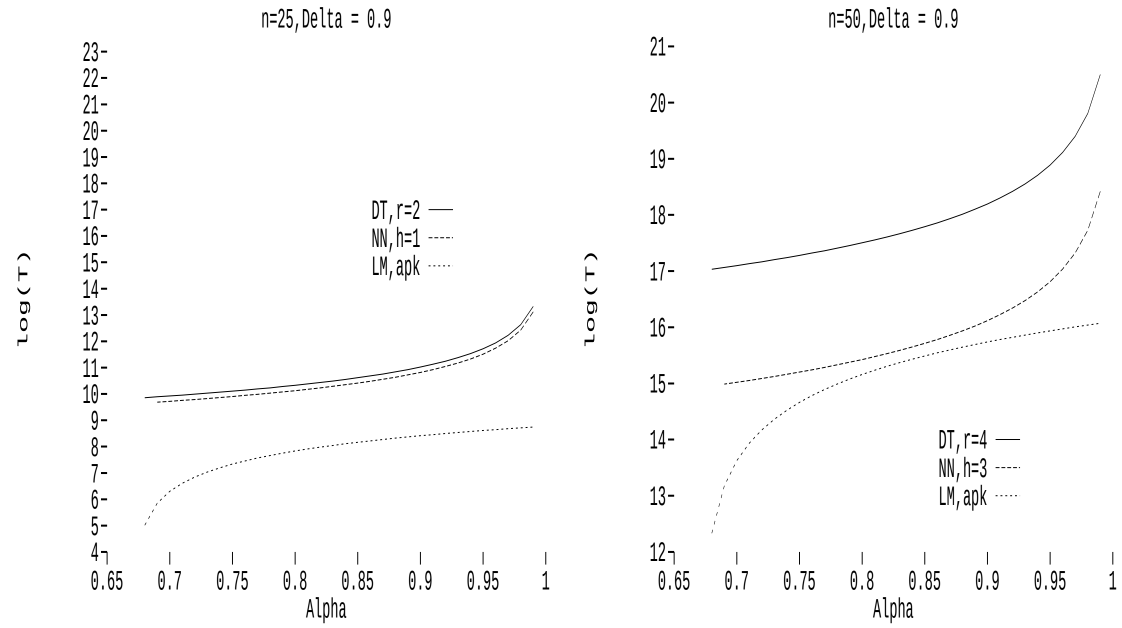

Figure 4: Accuracy α vs. $\log T$ for LM with a priori knowledge and $\delta = 0.9$ (a) n=25 and (b) n=50

to learn the class of all boolean concepts with at least accuracy α and confidence δ, were at the heart of this paper. By identifying possible sources of *impossibilities* the bounds suggested by the distinct sample model can be tailored to meet specifications of selected domains. This gives the model additional flexibility over, for example, the PAC model by allowing domain specific knowledge and domain encoding knowledge to be included. Several instances of domain information, such as rule knowledge, irrelevant attributes and values, and physical limitations were introduced. Finally, by means of 2 representative examples, the relative strength of the distinct sample model over 2 selected PAC bounds, were pointed out. Substantial savings in required samples were realized.

References

[1] Aha, D. W., Kibler, D., Albert, M. K. (1991), Instance-Based Learning Algorithms, Machine Learning, 6, Kluwer Academic Publishers, pp. 37-66.

[2] Baum, E. B. (1991), Neural Net Algorithms that Learn in Polynomial Time from Examples and Queries, IEEE Transactions on Neural Networks, Vol. 2, No. 1, January 1991, pp. 5-19.

[3] Baum, E., Haussler, D. (1989) What size net gives valid generalization. Neural Computation, 1, 1, pp. 151-160.

[4] Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, K. (1989) Learnability and the Vapnik-Chervonenkis Dimension, JACM, Vol. 36, No. 4.

[5] Clark, P., Niblett, T. (1987) Induction in Noisy Domains, Progress in Machine Learning, Ed. I. Bratko and N. Lavarac. Wilslow: Sigma Press.

[6] Cover, T.M., Hart, P.E. (1967) Nearest neighbor pattern classification, IEEE Transactions Information Theory, IT-13:21-27.

[7] Devroye, L.P. (1988) Automatic pattern recognition: A study of the probability of error, IEEE Trans. Pattern Analysis and Machine Intelligence, 10, 4, pp. 530-543.

[8] Ehrenfeucht, A., Haussler, D., Kearns, M., Valiant, L. (1988), A General Lower Bound on the Number of Examples Needed for Learning, COLT'88, Eds. D. Haussler and L. Pitt, MIT, August 1988, pp. 139-154.

[9] Ehrenfeucht, A., Haussler, D. (1988), Learning Decision Trees from Random Examples, COLT'88, Eds. D Haussler and L Pitt, MIT, August 1988, pp. 182-194.

[10] Haussler, D. (1988) Quantifying inductive bias: AI learning algorithms and Valiants learning framework. Artificial Intelligence 36, pp. 177-221.

[11] Kearns, M., Shapire, Sellie (1992) Towards Efficient Agnostic Learning, COLT'92.

[12] Laird, P.D. (1988), Learning from Good and Bad Data, Kluwer Academic Publishers, Boston, MA.

[13] Littlestone, N. (1988) Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. Machine Learning 2, 2, pp. 285-318.

[14] Perez, R.A., Hall, L.O., Romaniuk, S.G., Lilkendey, J,T. (1992), Inductive Learning For Expert Systems In Manufacturing, HICCS-25, Hawaii International Conference on Systems Sciences, Koloa, Hawaii, January.

[15] Porter, B.W., Bareiss, R., Holte, R.C. (1990) Concept Learning and Heuristic Classification in Weak-Theory Domains, Artificial Intelligence, Vol. 45, Nos. 1 and 2, pp. 710-746.

[16] Quinlan, J.R. (1979) Discovering rules by induction from large collections of examples. In D. Mitchie (Ed.), Expert systems in the micro electronic age. Edinburgh University Press.

[17] Quinlan, J. R. (1986), The effect of noise on concept learning, Machine Learning II, Los Altos, Morgan Kaufmann, 1986.

[18] Shakelford, G., Volper, D. (1988) "Learning k-DNF with Noise in the Attributes, COLT'88, Eds. D. Haussler and L. Pitt, MIT, August 1988, pp. 97-103.

[19] Sloan, R. (1988) Types of Noise in Data for Concept Learning, COLT'88, Eds. D. Haussler and L. Pitt, MIT, August 1988, pp. 91-96.

[20] Valiant, L.G. (1984), A Theory of the learnable, Comm. Ass. Comput. Mach., Vol. 27, No. 11, pp. 1134-1142.

[21] Vapnik, V.N., Chervonenkis, A. (1971) On the uniform convergence of relative frequencies of events to their probabilities. Theoretical Probability and Its Application, 16, 2, pp. 264-280.