The Exact Sample Complexity of PAC-learning Problems with Unit VC Dimension

Paul W. Goldberg
Neural Computing Research Group
Dept. of Computer Science and Applied Mathematics
Aston University
Birmingham B4 7ET
U.K.
pwgoldb@cs.sandia.gov

December 1996

Abstract

The Vapnik-Chervonenkis (VC) dimension is a combinatorial meansure of a certain class of machine learning problems, which may be used to obtain upper and lower bounds on the number of training examples needed to learn to prescribed levels of accuracy. Most of the known bounds apply to the Probably Approximately Correct (PAC) framework, which is the framework within which we work in this paper.

For a learning problem with some known VC dimension, much is known about the order of growth of the sample-size requirement of the problem, as a function of the PAC parameters. The exact value of sample-size requirement is however less well-known, and depends heavily on the particular learning algorithm being used. This is a major obstacle to the practical application of the VC dimension. Hence it is important to know exactly how the sample-size requirement depends on VC dimension, and with that in mind, we describe a general algorithm for learning problems having VC dimension 1. Its sample-size requirement is minimal (as a function of the PAC parameters), and turns out to be the same for all non-trivial learning problems having VC dimension 1. While the method used cannot be naively generalised to higher VC dimension, it suggests that optimal algorithm-dependent bounds may improve substantially on current upper bounds.

1 Introduction

The Vapnik-Chervonenkis (VC) dimension is a measure of a machine learning problem which gives bounds on its sample-size requirements. In the context of machine learning, the VC dimension was first applied by Blumer et al. [2] in the Probably Approximately Correct (PAC) learning framework. Subsequently it has been applied to the strongly related prediction framework in [7], and to a Bayesian framework in [6]. Both of the latter frameworks have given rise to algorithm-dependent bounds on sample-size that are very close to optimal. For the prediction framework, a result in [7] allows bounds obtained for that framework to be translated into PAC bounds. However the translation of the prediction bounds into PAC bounds is inefficient. It does however still give the best-known bounds on the order of growth of sample size as a function of the PAC parameters, although it does not quite match the lower bound of Ehrenfeucht et al. [3]. But the constant (ignored by an order-of-growth bound) is poor. It can be shown to imply more than 36 times the sample-size requirement of our algorithm, when applied to concept classes having VC dimension 1.

Other upper bounds on sample-size in the PAC framework [2, 10] are algorithm-independent, and hence cannot be expected to be better than [7], which is algorithm-dependent. An algorithm-independent bound is one that holds for any algorithm that finds a consistent concept. (The bound of [10] is currently the best known algorithm-independent bound on sample-size requirement in terms of VC dimension, in the PAC setting.) An algorithm-dependent bound is one that holds for some particular algorithm, whose hypothesis need not be a member of the concept class being learned. The gap between algorithm-dependent bounds and algorithm-independent bounds is real; an example in [7] shows that the sample-size requirement may grow as $\Theta(\epsilon^{-1} \log \epsilon^{-1})$, while their algorithm-dependent bound grows as $\Theta(\epsilon^{-1})$, as a function of the error bound ϵ. In section 3 we exhibit a version of that example.

Practitioners such as Holden and Niranjan [8] have found that all the known upper bounds on sample size requirement (also known as sample complexity) are too pessimistic, that is they overestimate the true sample complexity. This problem of overestimation is discussed in detail by Goldberg [4]. In this paper we show that for concept classes having VC dimension 1, in the PAC framework (described below), the sample complexity is $\log_{1-\epsilon} \delta$, which is the number of examples needed to sample a fixed ϵ-probable subset of the domain, with probability $1-\delta$. (This holds for all but "trivial" concept classes which have sample complexity of just one example.) We discuss the significance of the result after the basic definitions below.

Basic Definitions and Terminology

We give a brief description of the PAC learning framework and the terminology we will use. Further detail can be found in textbooks such as [1]. Examples are members of a set X called the

domain, and they are drawn at random from X by a fixed but unknown probability distribution P. Each example is given a binary label, ie. 0 or 1, and the labelling is assumed (in the basic PAC framework) to be performed by a deterministic function from X to $\{0,1\}$. So the function can be treated as a set $C \in X$, the set of members of X labelled by 1. This set is called the *target concept*, and belongs to a known *concept class* $\mathcal{C} \in 2^X$. $\mathcal{C}$ is known to the learner. So members of $\mathcal{C}$ are subsets of X and are called *concepts*; usually not all subsets of X are valid concepts, or the set X would have to be sampled almost exhaustively before much was known about C.

A learning algorithm returns a *hypothesis* $H \subseteq X$, which is intended to be good approximation to C. The error of H is the probability that a random example (drawn using P) is labelled differently by H and C. The PAC learning framework specifies two parameters, ϵ and δ, which specify respectively an upper bound on the error of H, and an upper bound on the probability of failing to attain that error bound. We say that H is ϵ-good if it has error $\leq \epsilon$. We want to know how many labelled examples we need to see (as a function of ϵ and δ) in order to learn any target concept from $\mathcal{C}$, regardless of the probability distribution P.

The Vapnik-Chervonenkis (VC) dimension is a measure of the sample complexity of a concept class. It is defined as follows. The VC dimension of $\mathcal{C}$ is the size of the largest possible subset $S \subseteq X$ having the property that for any binary labelling of the members of S, there exists a concept in $\mathcal{C}$ which labels the members of S accordingly. Such a set S is said to be *shattered* by $\mathcal{C}$.

Significance of the Result

The limitation of the applicability of the algorithm to concept classes with VC dimension 1 is quite restricted. What makes the result interesting is that it shows that for all concept classes with VC dimension 1, an optimal algorithm (with the usual assumption of no restriction on the unknown input distribution P) has the same sample complexity, namely $\log_{1-\epsilon} \delta$. This holds despite the (perhaps surprising) structural diversity of such concept classes. The only exception is "trivial" concept classes, where the sample complexity is 1, regardless of δ and ϵ. So the result suggests that VC dimension may be quite an accurate measure of sample complexity, and that good algorithm-dependent bounds may exist for higher values of VC dimension.

The structural diversity of concept classes with VC dimension 1 is elucidated in the next section, since the operation of the algorithm is conditioned on the "structural form" of whatever concept class is being learned. It is this feature which makes the approach apparently hard to extend to higher VC dimension, since it will be apparent that a complete characterisation of the kind we give for VC dimension 1 is infeasible for higher VC dimension.

2 Main Result

Let $\mathcal{C}$ be a concept class having VC dimension 1, with input domain X. We will start by assuming that X (and hence $\mathcal{C}$) is finite, then we will extend the result to the infinite case. The general approach is to give a structural taxonomy for concept classes of VC dimension 1. We believe that this approach is infeasible for higher VC dimension, due to the considerable increase in structural diversity of concept classes having VC dimension 2 (or higher). The algorithm we exhibit is then defined separately for the different structure classes.

First of all, if X contains any members a which are labelled in the same way by all members of $\mathcal{C}$ (ie. for all C_1, $C_2 \in \mathcal{C}$ we have $a \in C_1$ iff $a \in C_2$) then clearly those elements should be given the appropriate label by the algorithm. We may set aside these elements and focus attention on the others. Let $X' \in X$ denote the set of all elements whose labels are independent of the choice of target concept from $\mathcal{C}$.

We now observe that any pair of elements $a, b \in X$ may be labelled in up to 3 ways by members of $\mathcal{C}$. (If they were labelled in all 4 ways, they would be shattered, contradicting our assumption that $\mathcal{C}$ has VC dimension 1.) We first consider the case that they may be labelled in only 2 distinct ways by members of $\mathcal{C}$. In this case, if the labellings differ on just one member of the pair (for example $\langle a, b\rangle$ being labelled $\langle 0, 0\rangle$ or $\langle 0, 1\rangle$), then the other member of the pair is in X'. For neither a nor b to be in X', their labels may be $\{\langle 0, 0\rangle, \langle 1, 1\rangle\}$ or $\{\langle 0, 1\rangle, \langle 1, 0\rangle\}$. In this case we note that the label of a determines the label of b (and vice versa). The members of $X \setminus X'$ may be divided into equivalence classes of members of X whose labels are mutually determined.

Trivial Concept Classes

A trivial concept class is defined to be one for which the labels of all members of X are mutually determined, so that there is only one of the above equivalence classes. Clearly only one training example is needed to learn the target concept perfectly (with $\epsilon = \delta = 0$). $X' = \emptyset$ for trivial concept classes, and there are just 2 concepts.

Non-trivial Concept Classes

From now on we consider non-trivial concept classes, which have the property that there must exist $a, b \in X$ and two concepts in $\mathcal{C}$ which assign the same label to a and different labels to b. We show first that $\log_{1-\epsilon} \delta$ is a lower bound on sample complexity. Suppose that the input distribution P assigns probability $1 - \epsilon$ to a and ϵ to b. A learning algorithm must choose some label for b in the event that the training sample consists of repeated observations of the label of a. The target concept which assigns the opposite label to b will require a training set of size $\log_{1-\epsilon} \delta$, which is the sample size required to sample b with probability $1 - \delta$ (and thus correctly

label b).

We now show by construction of an appropriate algorithm that $\log_{1-\epsilon}\delta$ is an (algorithm-dependent) upper bound. The algorithm treats equivalence classes of elements of X whose labels are mutually determined, by labelling them in a consistent manner. We now choose a representative of each class, so that our domain has the property that every pair of elements has exactly 3 distinct labellings induced by the concepts. Any training example may be interpreted as a labelling of its class representative. From now on we will assume that any pair of elements has exactly 3 distinct labellings.

Some definitions:

For $a, b \in X$ we say that b *follows* a (denoted $b \succ a$) if for all $C \in \mathcal{C}$, $b \in C \Rightarrow a \in C$. That means that the labellings for $\langle a, b\rangle$ do not include $\langle 0, 1\rangle$. $\succ$ defines a partial order in X. If it is not the case that $a \succ b$ or $b \succ a$ then we say that a and b are *incomparable* under $\succ$. If a and b are incomparable under $\succ$ we say that a and b *compete for membership* if their labellings do not include $\langle 1, 1\rangle$, and they *compete for non-membership* if their labellings do not include $\langle 0, 0\rangle$.

Some Useful facts:

Fact 1: If $b \succ a$ and $b' \succ a$ with b and b' incomparable, then b and b' compete for membership. This is because when a is labelled 0, both b and b' must be labelled 0. So they do not compete for non-membership, hence they must compete for membership (or they would be shattered). Similarly, if $b \succ a$ and $b \succ a'$ with a and a' incomparable, then a and a' compete for non-membership.

Fact 2: $\succ$ cannot contain a loop, that is we cannot have distinct $a, b, b', c \in X$ with $c \succ b \succ a$ and $c \succ b' \succ a$ with b and b' incomparable. Since $b \succ a$ and $b' \succ a$, they must compete for membership (from fact 1). But their relationship with c implies that they also compete for non-membership, impossible under the assumption that any pair of elements have 3 distinct labellings.

Fact 3: If a competes with b for membership and with c for non-membership, then $b \succ c$. This inplies that if $a, b, c \in X$ are all mutually incomparable then either each pair of them competes for membership, or each pair of them competes for non-membership. Hence by extension, any set of at least 3 members of X that are mutually incomparable must either all compete for membership, or all compete for non-membership.

The $\succ$ Graph:

Given a concept class with VC dimension 1, we construct a directed graph on the representatives of equivalence classes of $X \setminus X'$ (the equivalence relation being the label determining equivalence

described above) as follows. We draw an edge from element b to element a if $b \succ a$ and there is no c with $b \succ c \succ a$. Since there are no directed cycles (indeed by fact 2 no cycles of any sort) it is convenient to assume that all edges are directed downwards, so that b appears above a. So, whenever $b \succ a$ there will be a path descending from b to a.

Terminology: A *monotonic path* is a path in a component of the $\succ$ graph that contains no incomparable elements. Connected components are trees, and usually they are rooted either at the top (with all paths directed from the root) or at the bottom (with all paths directed towards the root). We additionally define a *bi-tree* to be two trees with roots at the top and the bottom, which are joined at the root. We use the term "bi-tree" to refer to what could be called a "bi-forest", namely two sets F_1 and F_2 of trees, with members of F_1 rooted at the base and the top respectively, and a directed clique of edges descending from the roots of F_1 to the roots of F_2. The reason why this is essentially equivalent to a bi-tree is that the members of F_1 and F_2 could be connected via a common dummy root, which is sampled with probability 0. The set of concepts consistent with the structure is the same.

The *top half* of a bi-tree A denotes the set of all its vertices which compete for membership with some other vertex in A. The *bottom half* of A denotes the set of all vertices which compete for non-membership with some other member of A. The *trunk* of A refers to the (possibly empty) path connecting the top half with the bottom half.

Fact 4: Suppose that the $\succ$ graph for X contains a bi-tree. Then any concept in $\mathcal{C}$ contains either the bottom half of A minus a monotonic path descending from the trunk, or else the bottom half of A plus a monotonic path ascending from the trunk. Since incomparable elements of the top half compete for membership, a concept cannot contain anything more general than a monotonic path ascending from the trunk, and similarly since incomparable elements of the bottom half compete for non-membership, nothing more general than a monotonic path may be deleted fron the bottom half of A.

Fact 4 associates any concept in $\mathcal{C}$ with a unique element of a bi-tree in the $\succ$-graph, namely the one which defines the end of the path described above. (That may be the dummy element if the path is empty, and the concept contains just the bottom half elements.) We call this the *threshold element* for the bi-tree and concept.

A Taxonomy of Concept Classes with VC Dimension 1:

We construct the algorithm with reference to a case analysis on the $\succ$ graph, based primarily on the number of connected components that it has.

Case 1: The $\succ$ graph has more than 2 connected components.

Let A, B, C be 3 connected components, having members a, b, c respectively. a, b, c are mutually

incomparable, so by fact 3 either they all compete for membership, or else they all compete for non-membership. Assume the former (by symmetry the latter is equivalent).

For any element $d \in X$, assume without loss of generality that d is not in component A or B. So d, a and b compete for membership, and by the extension of fact 3, d competes for membership with any element of X which is incomparable with d. So, all incomparable elements compete for membership. So in this case a concept cannot contain elements from more than one connected component.

Now, for 3 elements a, a', b all in the same component with $b \succ a$ and $b \succ a'$, by fact 1 a and a' compete for non-membership. However this possibility has just been ruled out in the case under consideration. So the structure of each connected component is a tree (from fact 2) rooted at the base (under our convention of directing all edges down). This means that each member of $\mathcal{C}$ is a path within one of the connected components, with one end at the root of that connected component, which ascends monotonically to some node in the component.

In this case, out algorithm chooses as its hypothesis the smallest path corresponding to a function in the class that contains all members of X with label 1. This will be the empty set if no samples are labelled 1 (a set which might not belong to $\mathcal{C}$). An ϵ-good hypothesis is found provided that the ϵ-probable top section of the path is sampled, which gives the claimed sample complexity.

Case 2: The $\succ$ graph has 2 connected components.

Let A and B be the components. Either both components are monotonic paths, or there are 2 incomparable inputs in component A. Assume they compete for membership (the treatment is similar if they compete for non-membership). Then from fact 3 any incomparable inputs in component B compete for membership. Assume that there do in fact exist incomparable inputs in component B. Then any incomparable inputs in component A compete for membership. Then both components must be trees with roots at the base. If instead B has no incomparable inputs (it is a path) then A may be a bi-tree, defined above. But A may not be more general than this, ie. it may not contain vertices a, b, c where b and c compete for membership while a and b compete for non-membership. In this situation $c \succ a$ (the first observation in fact 1). But now let d be a member of B. Then a, b, d compete for non-membership while b, c, d compete for membership (using fact 3 both times). So b and d compete for both, a contradiction. We thus have:

Case 2a: As for case 1, but with only 2 connected components. The $\succ$-graph is two trees with roots at the bottom (or else with both roots at the top, if elements compete for non-membership). The algorithm works as before.

Case 2b: A is a bi-tree, B is a path. Members of the top half of A compete for membership

with members of B (by fact 3), so that if a concept contains a top-half element of A, it must contain no element of B. Similarly, if a concept does not contain some element of the bottom half of A then it must contain all elements of B. When the threshold element of A is in the trunk of A, the concept may contain some but not all of B. For two elements a, b in the trunk of A, with $b \succ a$, the set of all concepts associated with threshold b contain fewer elements of B than the concept associated with threshold a. Hence we can define an order on all concepts with thesholds in the trunk of B, in which concept C_1 precedes concept C_2 provided that either it contains fewer elements of B, or more elements of A.

Our algorithm works by choosing the first consistent hypothesis in the above ordering when applicable, else it chooses a minimum path in the top half of A, or minimal path deletion in the bottom half of A. We see that whetever the threshold elements in A and B, the condition for successful learning will be that some ϵ-probable set of elements is sampled with probability $1 - \delta$.

Case 3: The $\succ$ graph has 1 connected component.

Case 3a: Suppose first that there does not exist any $a \in X$ which competes for membership with some $b \in X$ and competes for non-membership with some $c \in X$. In this case the graph is a bi-tree, and the concept class is as described in fact 4.

The algorithm chooses a hypothesis that is as close as possible to the set consisting of the bottom half, a special case of case 2b (where there is an additional path).

Case 3b: Now suppose that there exists a as above. Let B be the subset of X that compete with a for membership, let C be the subset of X that compete with a for non-membership, and let A be everything else (including a). A is a bi-tree with a in the trunk. Everything in B follows everything in C, so B union C is also a bi-tree. There must be some edges between A and $B \cup C$, since there is only one connected component.

Let $A1$ be the followers of a in A, $A2$ be those followed by a in A. (So $A1$ and $A2$ contain the top half and bottom half of A respectively. There are no edges between $A1$ and B, since a and B compete for membership, and an edge between $A1$ and B would give a and a member of B a common follower f, and any concept containing f would contain both a and that element of B, a contradiction. Likewise there cannot be an edge between $A2$ and C. Hence all incomparable members of $A1$ and B compete for membership, and all incomparable members of $A2$ and C compete for non-membership.

Without specifying the edge between these bi-trees (there must be only one such edge since by fact 2 another edge would create a cycle), this shows that the concept class must be a subset of all sets of the following form: Construct a concept by starting with $A2 \cup C$ and add a path upwards from C (into B) and delete a path downwards from a. It is not possible to

delete a path downwards from a member of B since a is now a non-member and competes for non-membership with everything in B. Alternatively it is possible to add a path up from a and delete a path down from B. Not all of these concepts are allowed since they shatter pairs of elements of X from eg. B and $A2$. In general most members of $A1$ and B are followers of most members of $A2$ and C, but there may exist pairs $\{P_1, P_2\}$ of ascending/descending paths for which pairs $\{p_1, p_2\}$ of elements with $p_1 \in P_1$ and $p_2 \in P_2$ either compete for membership and non-membership. These paths may then be extended simultaneously, and there is a unique way of doing this so that one input is added at a time, which gives rise to an ordering on these pairs of paths.

As for case 3a, the algorithm chooses a hypothesis which is as close as possible to $A2 \cup C$. As before, if the target concept is anything other that $A2 \cup C$, then there is an ϵ-probable subset of X that must be sampled in order to guarantee that the hypothesis should be ϵ-good.

Extension to Infinite Concept Classes:

Suppose now that X is infinite. If $\mathcal{C}$ (having VC dimension 1) is finite then $\mathcal{C}$ divides X into finitely many (in fact $|\mathcal{C}|+1$) equivalence classes of elements which always receive the same label from elements of $\mathcal{C}$. Suppose that $\mathcal{C}$ is infinite. For input distribution P, let d be the metric on concepts which measures the expected absolute difference in label value on a point drawn at random under P. For a positive real number ζ, a ζ-cover of a metric space is defined to be a set of points in the metric space with the property that every point in the metric space is within distance ζ of some point in the ζ-cover. Pollard [9] provides a finite bound for the smallest possible ζ-cover of a concept class under the above metric in terms of the VC dimension and ζ. The size is independent of P, although obviously the ζ-cover itself may depend on P. For our purposes, the finiteness of the ζ-cover is sufficient.

Let $\mathcal{H}$ be a ζ-cover for $\mathcal{C}$. Thus there is a member $H \in \mathcal{H}$ that is within ζ of the target concept $C \in \mathcal{C}$. As $\zeta \to 0$ the probability that a random example is labelled differently by H and C approaches 0. Using H as the concept class, we can ensure that with probability $1 - \delta'$ there will be a consistent hypothesis (for the sample) in H, by choosing sufficiently small ζ, and δ' may be made arbitrarily small. This finite H thus gives us a PAC algorithm with error $\epsilon + \zeta$ and uncertainty $\delta + \delta'$. Since ζ and δ' may be arbitrarily small, the sample complexity can approximate the sample complexity for the finite case.

3 Further Observations and Conclusions

Obviously the main open problem we raise is the question of whether similar bounds exist for higher VC dimension. In particular, is it the case that "nearly all" concept classes of VC

dimension $d > 1$ have the same sample-size requirements for PAC learning? That would show that the VC dimension is really the right measure of complexity of concept classes, in the distribution-independent PAC setting. A result of that nature would have to be a result about optimal algorithm-dependent bounds, since we can exhibit a concept classes of VC dimension 1 for which an algorithm which is forced to find a consistent concept must have sample complexity which grows as $\Theta(\epsilon^{-1} \log \epsilon^{-1})$. The concept class, a variant of an example in [7], is the set of all single-element subsets of a set X of size ϵ^{-1}. For a target concept $C = \{x\}$ for some $x \in X$, let P assign probability 0 to x and probability $1/(\epsilon^{-1} - 1)$ to every other element. Then if a consistent *concept* must be returned as the hypothesis, it can be seen that the error is likely to be $1/(\epsilon^{-1} - 1)$ unless every class 0 example is sampled, which requires a sample size which grows as $\epsilon^{-1} \log \epsilon^{-1}$. Our optimal algorithm would make the hypothesis be the empty set, until a class 1 example was sampled.

So it would be interesting to know whether for VC dimension $d > 1$, there is a greater diversity of sample complexities for different concept classes of VC dimension d. The simplest concept class with VC dimension d is just the set of all subsets of a d-element domain. (For $d = 1$, this is a trivial concept class.) Such a concept class may always be augmented by the addition of an extra element whose label is the same for all concepts. The resulting concept class still has VC dimension d, but has higher sample complexity since P may now assign most probability to the new "uninformative" element of the domain. Could algorithm-dependent sample complexity for a concept class with VC dimension d depend only on whether or not there exist d shattered elements, together with an extra element whose label is the same for the shattering set of concepts? For $d = 3$, that property is held by circles in the plane, but not halfspaces in the plane. The class of circles in the plane is more general (since circles can approximate halfspaces), but it would be nice to know how much higher the sample complexity actually is.

An alternative and perhaps more appropriate route to getting good sample-size bounds may be to restrict the structure of concept classes under consideration. This suggestion is motivated by the kinds of concepts classes of VC dimension 1 that exist, not all of which are realistic, in the sense of corresponding to what might come up in practice. The kind of concept classes which cause problems, such as the one described in the first paragraph of this section, seem to involve large numbers of mutually exclusive elements. However, natural geometrical concept classes do not have that feature.

4 Acknowledgements

Research funded by EPSRC Grant Ref. GR/K 51792 *Validation and Verification of Neural Network Systems.* The author would like to thank Chris Bishop for drawing his attention to the general problem of the poor quality of sample-size bounds provided by VC dimension analysis, in the PAC learning framework.

References

[1] M. Anthony & N. Biggs (1992). *Computational Learning Theory.* Cambridge University Press.

[2] A. Blumer, A. Ehrenfeucht, D. Haussler, & M.K. Warmuth, (1989). Learnability and the Vapnik-Chervonenkis Dimension. *Journal of the Association for Computing Machinery*, 36(4), 929–965.

[3] A. Ehrenfeucht, D. Haussler, M. Kearns, & L.G. Valiant, (1989). A General Lower Bound on the Number of Examples Needed for Learning. *Information and Computation*, 82, 247–261.

[4] P. Goldberg (1996). The Gap Between Sample Size Requirements for Learning Problems and VC Dimension Based Bounds. *manuscript.*

[5] D. Haussler (1992). Decision Theoretic Generalizations of the PAC Model for Neural Net and Other Learning Applications. *Information and Computation*, 100, 78–150.

[6] D. Haussler, M. Kearns and R.E. Schapire (1994). Bounds on the Sample Complexity of Bayesian Learning Using Information Theory and the VC Dimension. *Machine Learning*, **14**, 83–113.

[7] D. Haussler, N. Littlestone and M.K. Warmuth (1994). Predicting $\{0,1\}$-Functions on Randomly Drawn Points. *Information and Computation* **115**, 248-292.

[8] S.B. Holden & M. Niranjan (1995). On the Practical Applicability of VC Dimension Bounds. *Neural Computation* **7**, 1265–1288.

[9] D. Pollard (1984). "Convergence of Stochastic Processes", Springer-Verlag, Berlin/New York.

[10] J. Shawe-Taylor, M. Anthony & N.L. Biggs (1993). Bounding sample size with the Vapnik-Chervonenkis dimension. *Discrete Applied Mathematics* **42** 65-73.

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

[12] L.G. Valiant (1985). Learning Disjunctions of Conjunctions. *Procs of the 9th International Joint Conference on AI,* pp. 560-566.

[13] V.N. Vapnik & A.Ya. Chervonenkis (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2), 264-280.