GA-based Learning of $kDNF_n^s$ Boolean Formulas

Arturo Hernández-Aguirre[1], Bill P. Buckles[1], and Carlos Coello-Coello[2]

[1] Department of Electrical Engineering and Computer Science, Tulane University, New Orleans, LA, 70118, USA
hernanda,buckles@eecs.tulane.edu
[2] CINVESTAV-IPN, Dept. de Ingeniería Eléctrica/Sección de Computación, México, D.F. 07300, MEXICO
ccoello@xalapa.lania.mx

Abstract. The number of samples needed to learn an instance of the representation class $kDNF_n^s$ of Boolean formulas is predicted using some tolerance parameters by the PAC framework. When the learning machine is a simple genetic algorithm, the initial population is an issue. Using PAC-learning we derive the population size that has at least one individual at some given Hamming distance from the optimum. Then we show that the population does not need to be close to the optimum in order to learn the concept.

1 Introduction

The size of the population is a critical factor for a Genetic Algorithm (GA). Although some theoreticians have favored the use of small populations [17], practitioners and early research indicate the opposite [4] It is normally accepted that a large population will converge at a slower pace but with a higher probability of success. "Small" and "large" are particular estimations derived from the many experimental populations tested on the problem.

In this paper, we seek to provide a characterization of the initial population through the estimation of the distance to the optimum. We apply the PAC-learning (Probably Approximately Correct learning) framework [19] to the estimation of an initial population that guarantees a fitness error between a hypothetical solution and one or more elements of the chromosome set. The metric is based on the Hamming distance formulation. Thus, the fitness function reports the number of bits by which two binary strings differ. A genotype is mapped onto itself to produce a phenotype. Therefore, the Hamming distance in the genotype domain equals that of the phenotype domain. In this way we can provide an environment in which the quality of a population size is estimated by the distance to the solution of the nearest chromosome. We call this size the PAC population.

The PAC framework sets a confidence δ over some predicted error ϵ, hence the terms approximately and correct. The PAC framework is commonly used to predict the generalization error of learning machines, such as neural networks [13, 22]. A careful reinterpretation of the PAC framework in the GA domain is developed throughout this paper. The class $kDNF_n^s$ (and several other classes) of Boolean formulas has been studied in the PAC framework as well. Thus, expressions for estimating sample complexity are well known and simple to use. We design a code suitable for representing Boolean formulas of the class. We show that the population used to learn a $kDNF_n^s$ concept is quite far from the optimum.

In this paper we develop an application of PAC-learning theory to genetic algorithms. The organization of this paper is as follows: First, we introduce (the simple) PAC-learning model. Then, we state the population size problem using the PAC-learning terminology, and we derive a GA sample complexity expression (population size). At the end, we derive the GA sample complexity for the binomial distribution case, and, we give our conclusions.

2 Definitions and previous work

Much of the initial work in PAC learning was devoted to find the sample complexity for learning a Boolean class. The class $kDNF$ is defined as follows (Kearns p.20 [11]):

$kDNF$: For any constant k, the representation class $kDNF_n$ consists of all Boolean formulae of the form $T_1 + T_2 + \ldots + T_l$, where each T_i is a conjunction of at most k literals over the Boolean variables $x_1, \ldots, x_n$.

The sample complexity for the class $kDNF_n^s$ (Kearns p.97 [11])

For fixed k, let $kDNF_n^s$ be the class of $kDNF$ formulae over $x_1, \ldots, x_n$ with at most s terms, and let A be a learning algorithm for $kDNF_n^s$. Then

$$S_A(\epsilon, \delta) = \Omega(\frac{1}{\epsilon} \ln \frac{1}{\delta} + \frac{s \ln \frac{n}{s}}{\epsilon}) \tag{1}$$

is the smallest sample complexity that upper bounds the error ϵ on future examples with confidence at least $1 - \delta$.

Pagallo [16] uses $S_A(\epsilon, \delta) = \frac{K \log(N)}{\epsilon}$ for learning the smallest DNF expression of the Boolean (target) concept. Iba et. al. [8, 9] use Pagallo formula to approach concept learning with four different techniques: neural networks, classifier systems, adaptive logic networks, and evolvable hardware. Based on the failure to learn the target concept in two out of three functions, they indicate ([8] p.14): "we can not conclude that evolvable hardware is superior to the other learning methods". They estimated sample complexity using a rather general formula, and they seem to oversee fundamental aspects of the PAC formalism. Their sample complexity expression assumes that the concept space and the hypothesis space are identical. That is, the classifier machine is assumed to produce only hypotheses in the (target) concept space. But, since their evolvable hardware system produces hypotheses in all the Boolean space (not only DNF expressions), one has to estimate (and work with) the VC-dimension of the hypotheses space spawned by the learning machine (instead of the VC-dimension of the target space). The VC-dimension of such evolvable hardware device would not be easy to estimate, but we can easily predict its size is enormous. The sample complexity would be larger than that reported by Pagallo's formula, thus easier to learn (in probability) [5].

The size of the initial population is the topic of papers by Goldberg [4], Mühlenbein & Schlierkamp-Voosen [14, 15], Reeves [17], and others. Cantú-Paz [3] worked out another estimation based on the *gambler's ruin model*. The model predicts the number of correct partitions in a chromosome at the time the convergence is attained.

In this paper we propose the distance to the optimum as a population quality measure (which is rather different than predict the final state of the population from some initial conditions). Thus, between two random populations we prefer the one with the closest element to the solution [6].

3 PAC-learning and GAs

The problem we address now is that of the population size of a genetic algorithm that contains, with probability greater than 1-δ one or more individuals at a distance no greater than ϵ (of a target hypothesis). By distance we mean the Hamming distance between two binary chromosomes. We use this metric as a measure of the error between two strings. A target hypothesis is anyone of the strings of length l that could encode a solution. PAC-learning uses the error and confidence parameters for deriving the size of the training set that guarantees a particular behavior of a learning machine. The size of this set is called *sample complexity*. Sample complexity is closely related to the population size of a genetic algorithm. In both cases (learning machines and GAs), the concept denotes a set of elements, sampled or generated from the solution space, with some fixed but possibly unknown probability distribution. PAC-learning usually assumes distribution-free data, which, in fact, is a general approach that covers the unknown distribution case.

3.1 PAC-learning concepts

PAC-learning is a theory of concept learning through examples [1, 20, 21]; we proceed as follows: assume a space $X \in \mathcal{R}$. A *concept class* $\mathcal{C} \subseteq X$ is the set of the possible partitions of X; a *hypothesis space* $\mathcal{H}$ is defined in the same way and may or may not be equal to $\mathcal{C}$; a target concept $c \in \mathcal{C}$ is a specific partition of X. Given a finite number of samples of a target concept c, the goal (of a learning machine) is to find the hypothesis $h \in \mathcal{H}$ which is a good approximation to c, with high probability (that is, with high probability, h follows very closely the behavior of c). The "good approximation" of h to c must be measured by some metric (or pseudo-metric at least) that estimates the error ϵ or symmetric difference between the hypothesis and the target concept. The "high probability" is indicated by the confidence parameter δ. More formally, for some small ϵ and δ, the hypothesis h satisfies

$$P_\mu[\ Error(h, c) > \epsilon] \leq \delta$$

for any target concept c and any probability of distribution P_μ. The minimum number of samples needed to attain that goal is called *sample complexity.*

The following example (taken from [7]) illustrates these ideas. We wish to learn the concept "medium build" from a set of examples taken from the space $X \in \mathcal{R}^2$. Every sample $\mathbf{x} = [x_1 x_2] \in X$ is a vector whose first component is "weight", and the second is "height". Let us assume the concept medium build is the area limited by the heights h_{max} and h_{min}, and the weights w_{max} and w_{min}, as shown in Figure 1.

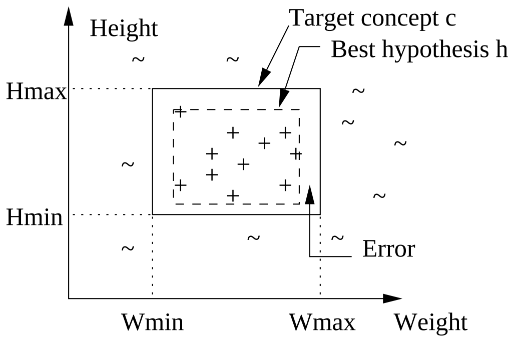

Fig. 1. The concept "medium build"

A set of examples is drawn from the space X and are classified by an *oracle* into either positive (represented by "+") or negative (represented by "$\sim$"). The best hypothesis is the smallest rectangle enclosing all positive examples. We can easily see that larger example sets might derive a better hypothesis, i.e., the closest hypothesis to the target concept.

3.2 Sample Complexity

Now we derive the sample complexity of a finite hypothesis space (the case of a GA). At this point we can estimate the error between the target and best hypothesis. Draw one more example $\mathbf{x} = [x_1 x_2] \in X$, and verify whether or not the sample falls inside the hypothesis rectangle (Figure 1). There is a small chance for an example to fall in the "error area". What is the probability of *not error*? It is at most $(1 - \epsilon)$ because ϵ is the chance of error. Now, if we tested m examples the chance of error becomes $(1 - \epsilon)^m$. We should assume a finite hypothesis space, thus, if all hypotheses have large error the probability that one of them correctly classifies the set X is at most $|\mathcal{H}| \cdot (1-\epsilon)^m$ ($|\mathcal{H}|$ is the cardinality of $\mathcal{H}$). Now, say we want to bound this probability by δ, we have:

$$|\mathcal{H}| \cdot (1-\epsilon)^m < \delta$$

Using the inequality $(1-x) < e^{-x}$, and later logarithms, the number of examples that bound the probability of misclassification by δ is:

$$m > \frac{1}{\epsilon}\big(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\big) \tag{2}$$

With that sample complexity, we can guarantee

$$P_\mu[\ Error(h,c) \le \epsilon] > 1-\delta \tag{3}$$

will hold for any new example drawn with the same probability of distribution [2]; this is the main PAC-learning equation.

3.3 Interpretation

We have suggested that sample complexity and population size of a GA are concepts of the same kind. A GA population of chromosomes is a population of coded candidate solutions. If the chromosomes are binary coded, and their length is n, the solution is one of the 2^n possible chromosomes; we name the solution chromosome the **target chromosome**.

Similarly, we name the finite number of chromosomes representable by 2^n, the **hypothesis space of chromosomes** $\mathcal{H}$. Therefore, each chromosome is a hypothesis in $\mathcal{H}$. The target chromosome is one such hypothesis which has the additional property of being a solution.

Let us concentrate in the initial population: how different are the initial individuals from the target chromosome? We measure the difference between two chromosomes by their **Hamming distance**. The Hamming distance is just another way to express the error ϵ, and we will find the exact relationship in the next section. Now, **what size must the initial population have to guarantee at least one hypothesis at a distance d from the target chromosome?** Figure 2 exemplifies the question. In an initial population, some individuals lie closer to the target hypothesis than others.

3.4 PAC-size population

In the following we modify Equation 2 for estimating the size of the GA initial population that with probability $1-\delta$ contains at least one individual at a Hamming distance d. For the sake of the discussion, assume the length of the binary strings is 20, $l = 20$. In the sample complexity formula, $m > \frac{1}{\epsilon}\big(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\big)$, we need to determine $\mathcal{H}$, ϵ and δ. The confidence δ is provided by the user and it does not depend on the problem domain.

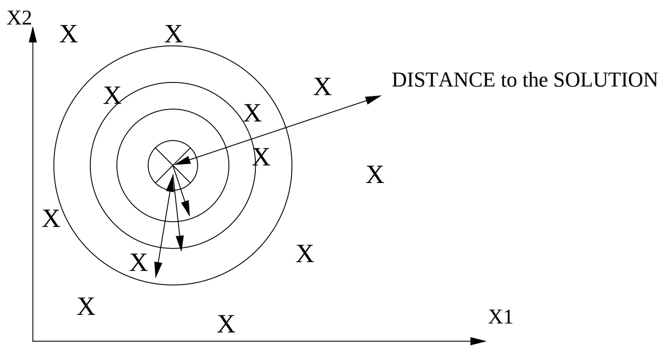

Fig. 2. Distance to the target hypothesis

The size of the hypothesis space of chromosomes is derived as explained in Section 3. The number of hypotheses representable by strings of length l is:

$$|\mathcal{H}| = 2^l$$

Substituting into Equation 2 and using logarithms, the sample complexity formula becomes:

$$m > \frac{1}{\epsilon}\left(l \ln 2 + \ln \frac{1}{\delta}\right) \tag{4}$$

The error ϵ needs a re-interpretation in the GA context. To any Hamming distance d corresponds an error ϵ. In Figure 1, the "Error" is the area of the target concept minus the area of the best hypothesis. Since the examples are uniformly distributed all over the plane, the area is proportional to the probability of error. The error associated to a Hamming distance $d = 0$, is the probability that $\binom{20}{0}$ out of the 2^{20} hypotheses, match the target hypothesis. This is, for example:

- $\epsilon(d = 0) = \frac{\binom{20}{0}}{2^{20}}$
- $\epsilon(d = 1) = \frac{\binom{20}{0}}{2^{20}} + \frac{\binom{20}{1}}{2^{20}}$
- $\epsilon(d = 2) = \frac{\binom{20}{0}}{2^{20}} + \frac{\binom{20}{1}}{2^{20}} + \frac{\binom{20}{2}}{2^{20}}$

For instance, the error associated with a Hamming distance of 2, $\epsilon(2)$, is the probability that one of $\binom{20}{0} + \binom{20}{1} + \binom{20}{2}$ hypotheses matches the target chromosome.
More formally, for binary strings of length l and Hamming distance d,

$$\epsilon(d) = \frac{\sum_{i=0}^{d} \binom{l}{i}}{2^l} \tag{5}$$

is the amount of error inherent to the Hamming distance. Equation 5 is part of Equation 4. In the following section we report some population size examples.

3.5 PAC-population examples

The computation of a PAC population is simple: we enter the desired Hamming distance into Equation 5 in order to determine the error, then the reported error and our desired level of confidence $(1 - \delta)$ (so, calculate δ) are entered into Equation 4 to determine the sample complexity or population size. The Table 1 shows PAC-population sizes for binary strings of length $l = 20$, for 10 Hamming distances, and 2 levels of confidence.

Error d	Confidence 90%	Confidence 95%
1	807181	841791
2	80336	83781
3	12547	13085
4	2736	2854
5	782	815
6	281	293
7	123	129
8	65	67
9	40	41
10	28	29

Table 1. The PAC-population with at least one element ϵ-close to a target chromosome

We have verified the accuracy of the confidence reported in Table 1 (we generated hundreds of random populations and measured the distance to a target chromosome). The predicted PAC-sized population was always correct. The explanation for this 100% accuracy is the following: we assumed a distribution-free sample complexity, but for the experiments each binary digit is generated with uniform probability. Therefore, the PAC-size is a loose upper bound to the population. In the following section we explain the exact solution to our problem. Since in our experiments we generate binary digits with uniform probability, the probability of having one chromosome in the population is exactly the same as for any other.

3.6 The binomial solution

We can find in our hypothesis space $\mathcal{H}$, a binary string of length l, and Hamming distance d from a target chromosome, with probability

$$p = \frac{\binom{l}{d}}{2^l}$$

where $|\mathcal{H}| = 2^l$. For a given confidence, and using the binomial distribution formula, we can say:

$$confidence = 1 - P[X = 0] = 1 - [\binom{m}{0} p^0 (1-p)^m]$$

$$confidence = (1-p)^m$$

Using logarithms, the "exact" population size is:

$$m = \frac{\ln(1 - confidence)}{\ln(1-p)} \qquad p = \frac{\binom{l}{d}}{2^l} \tag{6}$$

This expression is useful when we are aware of the density function. The Table 2 shows exact population sizes for binary strings of length $l = 20$, for 10 Hamming distances, and 2 levels of confidence.

Error d	**Confidence 90%**	**Confidence 95%**
1	120721	157062
2	12707	16532
3	2117	2754
4	498	647
5	156	202
6	62	80
7	30	39
8	18	24
9	14	18
10	10	16

Table 2. The "exact" population that provides at least one element ϵ-close to a target chromosome

When we contrast Tables 1 and 2 the increase in population due to the distribution-free assumption is apparent. In any case, we shall apply either formula according to the current conditions of our problem or experiments.

3.7 Non binary alphabets

In the previous sections we assumed binary alphabets, thus from there we generalize the results to alphabets of any cardinality. The sample complexity formula 2is not affected but error expression 5 involves the cardinality of the alphabet. Thus,

$$\epsilon(d) = \frac{\sum_{i=0}^{d} \binom{l}{i} * (R-1)^l}{R^l} \tag{7}$$

is the estimation of the error at a distance d for chromosomes of length l in alphabets of cardinality R. The Table 3 shows the PAC-population size for different distances and cardinality $R = 3$. Confidence $1 - \delta = 90\%$ in all cases.

The binomial solution for non-binary alphabets is the following:

$$m = \frac{\ln(1 - confidence)}{\ln(1-p)} \qquad p = \frac{\sum_{i=0}^{d} \binom{l}{i} * (R-1)^l}{R^l} \tag{8}$$

Error d	l=18	Error d	l=40
5	25895	23	321
6	5634	24	201
7	1530	25	136
8	510	26	99
9	206	27	77
10	99	28	64
11	57	29	56
12	38	30	52
13	29	31	49
14	25	32	48

Table 3. The PAC-population for cardinality $R = 3$ and confidence $\delta = 90\%$

The Table 4 shows the "exact"size for different distances and cardinality $R = 3$. Confidence is 90% in all cases.

Error d	l=18	Error d	l=40	Error d	l=180
5	3253	18	941	107	294
6	750	19	406	108	218
7	218	20	193	109	165
8	79	21	101	110	127
9	35	22	58	111	101
10	19	23	37	112	82
11	13	24	26	113	68

Table 4. The "exact" population for cardinality $R = 3$ and confidence $1 - \delta = 90\%$

4 GA learning $kDNF_n^s$ formulas

We use the GA as the machine to learn concepts of the $kDNF_n^s$ space. In the following three experiments, generalization error $\epsilon = 0.1$ and confidence $1 - \delta = 90\%$. The sample complexity is estimated using Equation 1. Phenotypes are encoded into genotypes through an alphabet of cardinality $R = 3$. A Boolean formula instance of the $kDNF_n^s$ space is: $f(x_1, x_2, x_3, x_4, x_5, x_6) = x_1x_2 + x_3x_4 + x_5x_6$. The parameters are: fixed number of literals $k = 2$, number of variables $n = 6$, and number of terms $s = 3$. The codification is as follows: each loci in the chromosome represents a variable, and the allele values are 0,1, and 2. A negated variable has allele 0, a positive variable allele 1, and allele 2 for a variable not present in the term. Therefore, the formula $f(x_1, x_2, x_3, x_4, x_5, x_6) = x_1x_2 + x_3x_4 + x_5x_6$ is encoded as: 112222221122222211 The string length is determined as $numberofterms \times numberofvariables$

The GA parameters are: binary tournament selection, $P_{crossover} = 0.7$, uniform crossover, $P_{mutation} = 0.2$ per individual.

4.1 Experiment 1

Learn the Boolean formula of 6 variables and 3 terms:

$$f = x_1\bar{x}_2x_3x_6 + \bar{x}_1x_3\bar{x}_5\bar{x}_6 + \bar{x}_3\bar{x}_4x_5x_6$$

The sample complexity is $S_A(\epsilon, \delta) = 44$. A population of 50 chromosomes finds a solution identical to the formula in generation 10 (of 20). Generalization of the Boolean expression was tested with a random population of 50 elements and has confidence greater than 98% (minimum estimated is 90%). The length of these chromosomes is 18, thus, in Table 4 we can find that the (estimated) closest element to the optimum lies at a Hamming distance between 8 and 9.

4.2 Experiment 2

Learn the Boolean formula of 10 variables and 4 terms:

$$\begin{aligned} f = {} & x_1\bar{x}_2x_3\bar{x}_4x_7x_8\bar{x}_9 + \bar{x}_1\bar{x}_2x_5x_6x_9x_{10} + x_1\bar{x}_2x_4\bar{x}_5x_6x_8x_9\bar{x}_{10} + \\ & x_1\bar{x}_2x_4\bar{x}_5x_6x_8x_9 + x_2\bar{x}_3\bar{x}_4x_6x_7x_{10} \end{aligned}$$

The sample complexity is $S_A(\epsilon, \delta) = 60$. A population of 70 chromosomes finds a solution identical to the formula in generation 12 (of 20). Generalization of the Boolean expression was tested with a random population of 512 elements ($\frac{2^{10}}{2}$) and has confidence greater than 95% (minimum estimated is 90%). The length of these chromosomes is 40. Thus, in Table 4 we can find that the (estimated) closest element to the optimum lies at a Hamming distance between 21 and 22.

4.3 Experiment 3

Learn the Boolean formula of 30 variables and 6 terms:

$$\begin{aligned} f = {} & x_1\bar{x}_2x_3\bar{x}_4x_7x_8\bar{x}_9\bar{x}_{11}\bar{x}_{12}x_{15}x_{16}x_{19}x_{20}x_{21}\bar{x}_{22}x_{24}\bar{x}_{25}x_{26}x_{28}x_{29}\bar{x}_{30} + \\ & x_2\bar{x}_3\bar{x}_4x_5x_6\bar{x}_8x_{10}x_{11}x_{12}\bar{x}_{13}\bar{x}_{16}\bar{x}_{18}x_{19}\bar{x}_{20}x_{21}x_{23}x_{24}x_{25}\bar{x}_{26}x_{27}\bar{x}_{29}x_{30} + \\ & \bar{x}_1x_4\bar{x}_5\bar{x}_6\bar{x}_7x_9\bar{x}_{10}x_{13}x_{14}\bar{x}_{15}x_{17}x_{18}x_{22}x_{23}\bar{x}_{27}x_{28}\bar{x}_{29}x_{30} + \\ & x_1x_2\bar{x}_3x_7x_8x_{11}x_{15}x_{19}\bar{x}_{20}\bar{x}_{21}x_{22}\bar{x}_{23}x_{24}x_{26}x_{28}x_{30} + \\ & \bar{x}_1x_4\bar{x}_5\bar{x}_6\bar{x}_7x_9\bar{x}_{10}x_{13}x_{14}\bar{x}_{15}x_{17}x_{18}x_{22}x_{23}\bar{x}_{27}x_{28}\bar{x}_{29}x_{30} + \\ & x_2x_5x_7x_9x_{10}x_{11}\bar{x}_{12}\bar{x}_{13}\bar{x}_{16}\bar{x}_{17}\bar{x}_{19}x_{20}\bar{x}_{22}\bar{x}_{24}x_{25}x_{26}x_{27} \end{aligned} \tag{9}$$

The sample complexity is $S_A(\epsilon, \delta) = 120$. A population of 120 chromosomes finds a solution with optimum fitness in generation 5 (of 20). This chromosome is different to the original formula in 112 locus (not alleles), and still the generalization is amazingly high. We tested with a random population of 100,000 elements (out of 2^30) to find a confidence greater than 98% (minimum estimated is 90%). The length of these chromosomes is 180. Thus, in Table 4 we can find that the (estimated) closest element to the optimum lies at a Hamming distance between 110 and 111.

5 Final remarks

We have shown the application of the PAC-learning formalism for estimating the population size of a GA that guarantees a distance requirement. Furthermore, we have made a new interpretation of the concepts to fit the GA paradigm.

It has been reported by several researchers that Evolvable Hardware (at gate-level) needs enormous computational resources [18, 12, 10]. Perhaps we have been approaching the problem in the incorrect way. In these experiments we can see how easy it is to find complex Boolean formulas. The key is to work in a hypotheses space whose instances are also members of the concept space. We show that in these conditions the GA does not need large populations and, in fact, the predicted best element of the initial population is quite far from the optimum. This is due to the proper representation of phenotypes, and of course, to the correct interpretation of the PAC theory that assumes a learning machine generating only hypotheses in the concept space.

Acknowledgments This paper describes research done in the Department of Electrical Engineering and Computer Science at Tulane University. Support for this work was provided in part by DoD EPSCoR and the Board of Regents of the State of Louisiana under grant F49620-98-1-0351. The third author acknowledges partial support from CINVESTAV through project JIRA'2001/08 and from the NSF-CONACyT project 32999-A.

References

1. M. Anthony and N. Biggs. *Computational Learning Theory.* Cambridge University Press, Cambridge, England, 1992.
2. Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis Dimension. *Journal of the ACM*, 36(4):929–965, October 1989.
3. Erick Cantú-Paz. *Efficient and Accurate Parallel Genetic Algorithms.* Genetic Algorithms and Evolutionary Computation. Kluwer Academic Press, 2000.
4. David E. Goldberg. Sizing Populations for Serial and Parallel Genetic Algorithms. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 70–79, San Mateo, California, 1989. Morgan Kaufmann Publishers.
5. Arturo Hernández-Aguirre. *Sample Complexity and Generalization in Feedforward Neural Networks.* PhD thesis, Department of Electrical Engineering and Computer Science, Tulane University, 2000.

6. Arturo Hernández-Aguirre, Bill Buckles, and Antonio Martínez-Alcántara. The pac population size of a genetic algorithm. In *Twelfth International Conference on Tools with Artificial Intelligence*, pages 199–202, Vancouver British Columbia, Canada, 13-15 November 2000. IEEE Computer Society.
7. S.B. Holden and P.J.W. Rayner. Generalization and PAC Learning: Some New Results for the Class of Generalized Single-layer Networks. *IEEE Transactions of Neural Networks*, 6(2):368–380, March 1995.
8. Hitoshi Iba, Masaya Iwata, and Tetsuya Higuchi. Machine Learning Approach to Gate-Level Evolvable Hardware. In Tetsuya Higuchi, Masaya Iwata, and Weixin Liu, editors, *Evolvable Systems: From Biology to Hardware. First International Conference (ICES'96)*, pages 327–343, Tsukuba, Japan, October 1996. Springer-Verlag.
9. Hitoshi Iba, Masaya Iwata, and Tetsuya Higuchi. Gate-Level Evolvable Hardware: Empirical Study and Application. In Dipankar Dasgupta and Zbigniew Michalewicz, editors, *Evolutionary Algorithms in Engineering Applications*, pages 259–276. Springer-Verlag, Berlin, 1997.
10. Tatiana G. Kalganova. *Evolvable Hardware Design of Combinational Logic Circuits*. PhD thesis, Napier University, Edinburgh, Scotland, 2000.
11. Michael J. Kearns. *The Computational Complexity of Machine Learning*. MIT Press, Cambridge, Massachusetts, 1990.
12. Julian F. Miller, Dominic Job, and Vesselin K. Vassilev. Principles in the Evolutionary Design of Digital Circuits—Part I. *Genetic Programming and Evolvable Machines*, 1(1/2):7–35, April 2000.
13. Tom Mitchell. *Machine Learning*. McGraw-Hill, Boston, Massachusetts, 1997.
14. Heinz Mühlenbein and Dirk Schlierkamp-Voosen. Predictive Models for the Breeder Genetic Algorithm, I: Continuous Parameter Optimization. *Evolutionary Computation*, 1(1):25–49, Spring 1993.
15. Heinz Mühlenbein and Dirk Schlierkamp-Voosen. The Science of Breeding and Its Application to the Breeder Genetic Algorithm (BGA). *Evolutionary Computation*, 1(4):335–360, Winter 1994.
16. Giulia Pagallo and David Haussler. Boolean Feature Discovery in Empirical Learning. *Machine Learning*, 5:71–99, 1990.
17. Colin R. Reeves. Using Genetic Algorithms with Small Populations. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 92–99, San Mateo, California, July 1993. University of Illinois at Urbana Champaign, Morgan Kaufmann Publishers.
18. Adrian Thompson, Paul Layzell, and Ricardo Salem Zebulum. Explorations in Design Space: Unconventional Design Through Artificial Evolution. *IEEE Transactions on Evolutionary Computation*, 3(3):167–196, September 1999.
19. Leslie G. Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.
20. Vladimir Naumovich Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, 1995.
21. Vladimir Naumovich Vapnik. *Statistical Learning Theory*. Wiley, New York, 1996.
22. M. Vidyasagar. *A theory of learning and generalization: with applications to neural networks and control systems*. Springer-Verlag, London, 1997.