Appears in the
Proceedings of the AAAI Fall Symposium on "Relevance",
New Orleans, November 1994.

Exploiting the Absence of Irrelevant Information: What You Don't Know *Can* Help You

R. Bharat Rao, Russell Greiner and Thomas Hancock*
Learning Systems Department
Siemens Corporate Research
755 College Road East
Princeton, NJ 08540-6632
{bharat, greiner, hancock}@learning.scr.siemens.com

Abstract

Most inductive inference algorithms are designed to work most effectively when their training data contain *completely specified* labeled samples. In many environments, however, the person collecting the data may record the values of only some of the attributes, and so provide the learner with only partially specified samples. This can be modeled as a *blocking* process that hides the values of certain attributes from the learner. While blockers that remove the values of critical attributes can handicap a learner, this paper instead focuses on blockers that remove only superfluous attribute values, *i.e.*, values that are not needed to classify an instance, given the values of the other unblocked attributes. We first motivate and formalize this model of "superfluous-value blocking", and then demonstrate that these omissions can be quite useful, showing that a certain class that is seemingly hard to learn in the general PAC model — *viz.*, decision trees — is trivial to learn in this setting. We also show how this model can be applied to the theory revision problem.

Introduction

Diagnosticians often perform tests sequentially and stop as soon as they know the correct diagnosis; they, therefore, typically perform only a small fraction of all possible tests. As an example, knowing that a positive t_1 blood test is sufficient to establish that a patient has `diseaseX`, a doctor can conclude that a patient has `diseaseX` after performing only t_1, if that test is positive. In recording his findings, the doctor will only record $\langle t_1, + \rangle$ and the diagnosis `diseaseX`, and will not record the patient's symptoms corresponding to the other tests $t_2, t_3, \ldots, t_n$.

A learning program may later examine the doctor's files, trying to learn the doctor's diagnostic procedure. These records are "incomplete", in that the values of many attributes are missing; *e.g.*, they do *not* include the results of tests t_2 through t_n for this patient. However, within our model, the learner can use the fact that these attributes are missing to conclude that *the missing tests are not required to reach a diagnosis, given the values of the other known tests*; which here means that the target classifier should establish `diseaseX` and terminate on observing only $\langle t_1, + \rangle$. This paper shows that such omissions can actually be beneficial, by presenting learning algorithms that can exploit these types of omissions to learn classifiers that appear hard to learn without this auxiliary information.

Motivation and Related Work: Existing learning systems that permit incomplete information in the learning samples [BFOS84; Qui92] are all based on a different learning model [SG94]: After one process draws completely-specified samples at random, a second process then hides the values of certain attributes, perhaps changing the complete tuple $\langle 1, 0, 1, 1 \rangle$ to the partial tuple $\langle *, 0, *, 1 \rangle$, where "$*$" means this value is hidden. From such partial tuples, each labeled with its correct class, the learner must produce a classifier that can classify unlabeled partial tuples. Here, the hiding process, called "blocking", can be viewed as a type of random noise.

The model in this paper also assumes a random process is generating complete attribute tuples, which are then partially blocked. Our model differs by dealing with blocking processes that are based, in a particular manner, on the values of the attributes and on the target classifier that the learner is trying to acquire: here, we can assume that the unblocked data alone is sufficient to establish the correct diagnosis. We view the values of the blocked attributes as "superfluous" or "conditionally irrelevant": *irrelevant* as they do not play a role in classifying the sample, given the values of the other specified values (*i.e.*, *conditioned* on those other values). *E.g.*, as the doctor will conclude `diseaseX` if t_1 is positive, whether t_2 is positive or negative, we say that "t_2 is *superfluous*, given that t_1 is positive". Of course, if t_1 is

*Authors listed alphabetically by first name. We gratefully acknowledge receiving helpful comments from Dale Schuurmans and George Drastal.

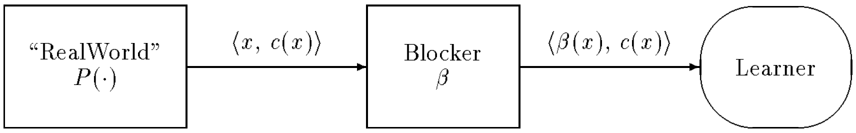

Figure 1: Model of Blocking Process

negative, t_2 may then be relevant for the diagnosis. Most other research that deals with learning in the context of irrelevant attributes [Lit88; Blu92; AD91; BHL91] considers the stronger claim that an attribute is irrelevant only if its value *never* plays a role in the classification, for any values of the other features.[1]

Implemented learning systems that can handle missing features tend to work effectively when relative few feature values are missing, and when these missing values are randomly distributed across the samples. However, recent studies [PBH90; RCJ88] have shown that in practice many datasets are missing more than half of the feature values! Moreover, these attribute values are not randomly blocked, but in fact "*are missing [blocked] when they are known to be irrelevant for classification or redundant with features already present in the case description*" [PBH90], which is precisely the situation considered in this paper (see Definition 2). Our model of learning may therefore be applicable to many diagnostic tasks, and will be especially useful where experts are not available, or are unable to articulate the classification process they are using.

Framework

Standard Model of Learning: Following standard practice, we identify each domain instance with a finite vector of boolean attributes $x = \langle x_1, \ldots, x_n \rangle$, and let $X_n = \{0,1\}^n$ be the set of all possible domain instances. The learner is trying to learn a concept which we view as an indicator function $c : X_n \mapsto \{T, F\}$, where x is a member of c iff $c(x) = T$.[2] A "(labeled) example of a concept $c \in \mathcal{C}$" is a pair $\langle x, c(x) \rangle \in X_n \times \{T, F\}$. We assume that random labeled instances are drawn independently from a stationary distribution $P : X_n \mapsto [0, 1]$.

A learning algorithm has access to an oracle, EX, for random examples that, when called, will provide (in unit time) a labeled example $\langle x, c(x) \rangle$ where x is drawn according to P and labeled by c. The learning algorithm's goal is to produce as output a concept h that has small error, where the error of h is defined as

$$Err(h) \quad = \sum_{x \in X_n, h(x) \neq c(x)} P(x)$$

which is the probability that the returned hypothesis h will misclassify an instance x drawn from the distribution P. We use the standard PAC criterion [Val84; KLPV87] to specify the desired performance of our learners:

Definition 1 (PAC learning) *Algorithm* L PAC learns *a set of concepts* $\mathcal{C}$ *if, for some polynomial function* $p(\cdots)$*, for all target concepts* $c \in \mathcal{C}$*, distributions* P*, and error parameters,* $0 < \epsilon, \delta < 1$*, when* L *is run with inputs* ϵ*,* δ*, and a random example oracle* EX *(where each call to* EX *returns an instance independently chosen according to distribution* P *and labeled according to* c*),* L *terminates in time at most* $p(\frac{1}{\epsilon}, \frac{1}{\delta}, |c|)$*, where* $|c|$ *is the "size" of the concept* c*, and outputs a hypothesis* $h \in \mathcal{C}$ *whose error is, with probability at least* $1 - \delta$*, at most* ϵ*; i.e.,*

$$\forall P \in \text{“distr. on } X_n\text{”},\ c \in \mathcal{C},\ 0 < \epsilon, \delta < 1,$$
$$\Pr(\ \mathrm{Err}(h) < \epsilon\ ) \quad \geq \quad 1 - \delta\ .$$

Model of "Blocked Learning": In standard learning models, each labeled instance $\langle x, c(x) \rangle$ is passed "as is" to the classifier. In our model the learner instead only sees a "censored" version of $\langle x, c(x) \rangle$, written "$\langle \beta(x), c(x) \rangle$", based on a blocker β which is a function that maps an instance $x \in \{0,1\}^n$ to a blocked instance $\beta(x) \in \{0, 1, *\}^n$; see Figure 1. The value "$*$" denotes the the corresponding attribute is blocked. Hence, a blocker could map $\langle \langle 1,1,0,1 \rangle, T \rangle$ to $\langle \langle 1,*,*,* \rangle, T \rangle$, or $\langle \langle *,1,0,* \rangle, F \rangle$, but not to either $\langle \langle 1,0,0,1 \rangle, T \rangle$ or $\langle \langle 1,1,0,1 \rangle, * \rangle$. Let $X_n^* = \{0, 1, *\}^n$ denote the set of possible instance *descriptions.*

This paper considers only *superfluous-value blockers*: *i.e.*, blockers that block only attribute values that do not affect an instance's classification, given the values of the other unblocked attribute values. (Therefore, for superfluous value blockers, "$*$" is equivalent to a "don't care" value.) To state this more precisely:

Definition 2 ("Superfluous"/"Conditionally Irrelevant") *Given a concept* $c \in \mathcal{C}$ *over the set of attributes* $\{x_1, \ldots, x_n\}$*:*

1. *The attributes* $\{x_{m+1}, \ldots, x_n\}$ *are* superfluous *given the partial assignment* $\{x_1 \mapsto v_1,\ \ldots,\ x_m \mapsto v_m\}$ *(where each* $v_i \in \{0,1\}$ *is a specified value) iff*

[1] Other research [JKP94] extends the traditional notion of irrelevance, to handle correlated attributes.

[2] To simplify our presentation, we will assume that each attribute has one of only two distinct values, $\{0,1\}$, and that there are only two distinct classes, written $\{T, F\}$. It is trivial to extend this analysis to consider a larger (finite) range of possible attribute values, and larger (finite) set of classes.

$\forall i = (m+1)..n,\ y_i \in \{0,1\}$:
$c(\langle v_1, \ldots, v_m, y_{m+1}, \ldots, y_n\rangle) = c(\langle v_1, \ldots, v_m, 0, \ldots, 0\rangle)$

2. *a function* β: $X_n \mapsto X_n^*$ *is a legal* superfluous-value blocker *if it only blocks superfluous attributes; i.e., if* $\beta(\langle v_1, \ldots, v_n\rangle) = \langle v_1, \ldots, v_m, *, \ldots, *\rangle$ *implies the attributes* $\{x_{m+1}, \ldots, x_n\}$ *are superfluous given the partial assignment* $\{x_1 \mapsto v_1,\ \ldots,\ x_m \mapsto v_m\}$.

To better clarify our notion of "superfluous", consider the situation where $X_n = \{0,1\}^2$ is the set of all possible domain instances and the target concept c is represented by the boolean formula $x_1 \vee x_2$. Then a legal blocker could map the instance $\langle\langle 1,1\rangle, T\rangle$ either to $\langle\langle 1,*\rangle, T\rangle$ or to $\langle\langle *,1\rangle, T\rangle$, meaning "$x_2$ is superfluous, given that x_1 is positive" and vice versa. Hence, for this single instance, either x_1 or x_2 can be conditionally irrelevant, depending on the tests that have already been performed. Therefore, our notion of *(ir)relevance* depends on both the instance being filtered and on the blocker that is employed.

In general, there can be many possible superfluous value blockers for a given concept (including the degenerate no-op blocker that reveals every attribute). We must therefore specify the blocking scheme $\mathcal{B}$ that maps each concept $c \in \mathcal{C}$ to its superfluous value blocker $\beta = \mathcal{B}(c)$. This paper focuses on blocking schemes which correspond to an evaluation procedure that takes an instance and a target concept, and runs until the value of the target is determined, leaving unblocked just those attributes that are examined by the evaluation procedure. For example, the natural blocking scheme for decision trees is one that evaluates an instance on the target tree and reveals only those attributes that were tested on the root–to–leaf path taken by the instance.

Our model is otherwise the same as the PAC model. In particular, the learner's task is still to acquire a classifier that has high accuracy on *completely specified* instances. (This objective is reasonable as the classifier can specify the tests to be performed. Recall, however, the learner only sees training instances that have been filtered through a legal blocker.)

We say that "a concept class $\mathcal{C}$ is learnable with respect to the blocking scheme $\mathcal{B}$" if there is a learning algorithm L that achieves the PAC criterion shown in Definition 1, with the sole modification that L uses the blocked-oracle $EX_{\mathcal{B}(c)}$, where on each call, $EX_{\mathcal{B}(c)}$ returns the labeled *blocked* instance $\langle \mathcal{B}(c)(x),\ c(x)\rangle \in X_n^* \times \{T,F\}$, rather than the unblocked $\langle x, c(x)\rangle \in X_n \times \{T,F\}$ that EX would have returned. (*I.e.*, $EX_{\mathcal{B}(c)}$ first filters the instance through the $\beta = \mathcal{B}(c)$ filter associated with target concept c.)

Results for Decision Trees

This section discusses the challenge of learning decision trees within this "superfluous blocking model". Decision trees have a natural blocking strategy since each example is classified by a unique root-leaf path in the tree. We consider the blocker that suppresses those attributes that do not appear on an instance's evaluation path in the target decision tree. We begin with the simplest situation, and progress to successively more complicated situations. For notation, let $\mathcal{DT}_n$ be the set of decision trees defined over n boolean variables, X_n.

Empty Initial Tree: The $DT[E]$ model assumes that the learner begins with an Empty initial decision tree, which is extended into a full decision tree based on a set of samples, and that the blocker uses the target decision tree d_T to block attributes of the instances. On receiving a complete instance, $x = \langle x_1, \ldots, x_n\rangle$, the blocker traverses the d_T decision tree, from the root down, recording not just the appropriate classification for this instance $d_T(x) \in \{T,F\}$, but also the set of tests that were performed; this corresponds to the path through d_T. The blocker then passes on only the attributes encountered on the path, and blocks all the other attributes.

The simple GrowDT$_s$ algorithm, shown in Figure 2, can learn decision trees from these blocked instances in the $DT[E]$ model. The critical observations are that the root of the target tree can never be blocked and that any variable that is never blocked appears on the path to every leaf reached by the sample and hence can be placed at the root.[3] After GrowDT has collected "enough" samples, it calls BuildDT to find the best tree, given those samples. BuildDT first selects as its root any attribute value that is never blocked, then splits on this attribute, and calls itself recursively.[4]

Theorem 1 *For any target tree* $c_0 \in \mathcal{DT}_n$*, any values of* $\epsilon, \delta \in (0,1)$ *and any distribution, under the* $DT[E]$ *model,* GrowDT*(* n*,* ϵ*,* δ *) will, with probability at least* $1-\delta$*, return a tree* $c' \in \mathcal{DT}_n$*, whose error is at most* ϵ*. Moreover,* GrowDT *requires* $O(|c_0|\frac{1}{\epsilon}\ln(\frac{n}{\delta}))$ *samples, and returns a tree of size* $|c'| \le |c_0|$*, where* $|c|$ *is the number of nodes in* c*.*

We see here the advantage of exploiting these superfluous values, as there is currently no known algorithm capable of learning decision trees in the standard PAC model.[5]

Non-Empty Initial Tree: In many situations, we may already have an initial decision tree, $d_{init} \in \mathcal{DT}_n$,

[3] If the blocker also provides additional information about the *order* in which tests are recorded, then GrowDT can be trivially modified to learn a tree identical to the d_T decision tree traversed by the blocker.

[4] While the GrowDT$_s$ algorithm is parameterized by the size of the tree s, we can produce a modified version GrowDT that does not require this parameter by using the standard technique [HKLW91] of repeated attempts with successively doubled estimates of s.

[5] The best known general algorithms run in pseudo-polynomial time, *i.e.*, they learn an s-node decision tree in time polynomial in $s^{\log s}$ [EH89].

```
Algorithm GrowDT_s( n: Z+, ε: (0,1), δ: (0,1) ): TreeType
    /* Returns tree of size s that correctly classifies a (large enough) sample of n-tuples */
  Draw   m_s = (1/ε)[s ln(8n) + ln(1/δ)]   partially-specified labeled samples, S_s (from the EX_B(c) oracle)
  Return( BuildDT( S_s ) )
End GrowDT_s

Algorithm BuildDT( S: set_of_partial_samples ): TreeType
          /* Builds a tree using samples S */
  Let NT be new tree
  If (all samples in S are labeled with same ℓ) or (S is empty) then
    NT.LeafLabel = ℓ    (or "... = T" if |S| = 0)
    Return( NT )
  Let x_i be any variable that is unblocked forall s ∈ S.
  Let S^0 = {⟨x − {x_i/0}, ℓ⟩ | ⟨x, ℓ⟩ ∈ S,  x_i/0 ∈ x}
      S^1 = {⟨x − {x_i/1}, ℓ⟩ | ⟨x, ℓ⟩ ∈ S,  x_i/1 ∈ x}
        /* To form S^0: Assemble the instances in S that include the x_i/0 assignment,
           then project out that attribute; similarly for S^1. */
  Let NT.InternalNodeLabel = x_i
      NT.If0 = BuildDT( S^0 );     NT.If1 = BuildDT( S^1 )
  Return( NT )
End BuildDT
```

Figure 2: BuildDT Algorithm for learning decision trees

which is considered quite accurate, but not perfect. We can use a set of labeled samples to modify d_{init}; *i.e.*, to form a new tree d_{better} that is similar to d_{init}, but (with high probability) more accurate. We let $DT[TR+]$ refer to this model, where the TR designates "Theory Revision", corresponding to the many existing systems that perform essentially the same task (typically for Horn-clause based reasoning systems) [MB88; OM90; Tow91; WP93; LDRG94].

As before, the real-world continues to generate completely-specified samples, x, each labeled $d_T(x) \in \{T, F\}$ by the target (but unknown) decision tree d_T. Here, however we need a slightly different type of blocker, which first runs this sample through the *initial decision tree* d_{init}, to obtain both a proposed (but not necessarily correct) label $d_{init}(x)$ and a particular set of unblocked attribute values; call this $\beta_{d_{init}}(x) \in X_n^*$. If d_{init}'s label is correct (*i.e.*, if $d_T(x) = d_{init}(x)$), then the learner receives $\langle \beta_{d_{init}}(x), d_T(x)\rangle$. Otherwise, if $d_T(x) \neq d_{init}(x)$, the learner receives $\langle \beta_{d_{init} \vee d_T}(x), d_T(x)\rangle$, where the $\beta_{d_{init} \vee d_T}$ blocker specifies the value of any attribute that is unblocked by either $\beta_{d_{init}}$ or β_{d_T}.

The extended paper [GHR94] presents the ModifyDT$^+$ algorithm (an extension of the earlier BuildDT algorithm) which can modify an initial decision tree d_{init}, based on a set of such labeled partially-specified instances. It also shows that ModifyDT$^+$ can effectively pac-learn the target decision tree.

Learner is also told which attributes were NOT needed: Alternatively, the blocker might also specify that some of the attributes that d_{init} included are in fact superfluous; *i.e.*, that some of the tests that were performed should not have been run. The $DT[TR\pm]$ model provides this information by using the β_{d_T} blocker. (As the learner knows d_{init}, it can determine which attributes $\beta_{d_{init}}$ would block, if necessary.) [GHR94] describes the ModifyDT$^\pm$ algorithm that can pac-learn decision trees given such instances.

Extending the "perfect blocker" model: Each of the blockers described above is "perfect", *i.e.*, will remove all-and-only the superfluous attribute values. In practice, however, a diagnostician may occasionally perform a superfluous test or omit a relevant test, or could even mis-read or mis-record some attribute values. These types of errors can occur, for example, if different patients are diagnosed by different experts, each of whom uses his own tree to do the classification. The extended paper [GHR94] generalizes our "perfect" model by removing these restrictions, producing various $\beta^{\vec{p}}$ blockers, each parameterized by the 6-tuple of probability values, $\vec{p} = \langle p_{rc}, p_{r*}, p_{ri}, p_{sc}, p_{s*}, p_{si}\rangle$, where

	Attribute value is	
Blocker's value is:	Required	Superfluous
Correct	p_{rc}	p_{sc}
*	p_{r*}	p_{s*}
Incorrect	p_{ri}	p_{si}

To explain: Given any target decision tree d_T, for each instance, we can label each attribute value as either required or superfluous. For each required attribute, $\beta^{\vec{p}}$ stochastically makes a 3-way decision: providing the attribute's correct value with probability p_{rc}, "*" with probability p_{rc}, and the incorrect value with probability p_{ri} (e.g., returning 0 when the correct attribute value is 1). Similarly, if the attribute is superfluous, $\beta^{\vec{p}}$ will return the attribute's correct value, the value "*" or the wrong value, with respective probabilities p_{sc}, p_{s*}, or p_{si}.

The $DT[E]$ model, discussed above, uses $p_{rc} = p_{s*} = 1$ and the other p_i values all 0; and the traditional complete-tuple model uses $p_{rc} = p_{sc} = 1$. We can use this framework to connect this work with earlier models of attribute noise; in particular, as [KL88; SV88] consider erroneous but not missing attribute values, they each assume $p_{r*} = p_{s*} = 0$.

As a final point, we have considered learning arbitrary DNF formulae when superfluous values are omitted. Here, each blocked instance is simply an implicant of either the target formula φ or its negation $\neg\varphi$. There are several different ways of specifying which implicant should be returned, including "the smallest prime implicant", and "the values of the smallest prefix of a single ordering, say $\langle x_1, \ldots, x_n \rangle$, that qualifies as an implicant". We present several situations where a benevolent blocker (think "teacher") can make formulae trivial to learn, and other situations where an adversarial blocker can make learning such formulae as difficult as learning DNF in the usual "completely-specified tuple" model.

Summary

Even though most diagnosticians perform and record only a small fraction of the set of possible tests to reach a classification, most learning systems are designed to work best when their training data contains completely-specified attribute-value tuples. This abstract presents learning algorithms designed to deal with exactly such partially-specified instances. We show, in particular, that it is easy to "PAC learn" decision trees in this model. We also present extensions to the underlying model, and suggest corresponding algorithms, to incrementally modify a given initial decision tree, and provide a framework for discussing various types of "noise" — where the blocker can stochastically omit required values, or include superfluous values.

References

H. Almuallim and T.G. Dietterich. Learning with many irrelevant features. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 547–552, 1991.

L. Breiman, J. Friedman, J. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

A. Blum, L. Hellerstein, and N. Littlestone. Learning in the presence of finitely or infinitely many irrelevant attributes. In *Proc. 4th Annu. Workshop on Comput. Learning Theory*, pages 157–166. Morgan Kaufmann, San Mateo, CA, 1991.

A. Blum. Learning boolean functions in an infinite attribute space. *Machine Learning*, 9(4):373–386, October 1992.

A. Ehrenfeucht and D. Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231–246, 1989.

Russell Greiner, Thomas Hancock, and R. Bharat Rao. Knowing what doesn't matter: Exploiting (intentionally) omitted superfluous data. Technical report, Siemens Corporate Research, 1994.

D. Haussler, M. Kearns, N. Littlestone, and M. K. Warmuth. Equivalence of models for polynomial learnability. *Inform. Comput.*, 95(2):129–161, December 1991.

G.H. John, R. Kohavi, and K. Pfleger. Irrelevant features and the subset selection problem. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 121–129, 1994.

M. Kearns and M. Li. Learning in the presence of malicious errors. In *Proc. 20th Annu. ACM Sympos. Theory Comput.*, pages 267–280. ACM Press, New York, NY, 1988.

Michael Kearns, Ming Li, Leonard Pitt, and Leslie Valiant. On the learnability of boolean formulae. In *Proceedings of the 19th Symposium on the Theory of Computations*, pages 285–295, New York, May 1987.

Pat Langley, George Drastal, R. Bharat Rao, and Russ Greiner. Theory revision in fault hierarchies. Technical report, SCR Techreport, Siemens Corporate Research, 1994.

Nick Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning Journal*, 2:285–318, 1988.

S. Muggleton and W. Buntine. Machine invention of first order predicates by inverting resolution. In *Proceedings of IML-88*, pages 339–51. Morgan Kaufmann, 1988.

Dirk Ourston and Raymond J. Mooney. Changing the rules: A comprehensive approach to theory refinement. In *Proceedings of AAAI-90*, pages 815–20, 1990.

B. W. Porter, R. Bareiss, and R. C. Holte. Concept learning and heuristic classification in weak-theory domains. *Artificial Intelligence*, 45(1-2):229–63, 1990.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, 1992.

K. Ruberg, S.M. Cornick, and K.A. James. House calls: Building and maintaining a diagnostic rule-base. In *Proceedings Third Knowledge Acquistion for Knowledge-Based Systems Workshop*, 1988.

Dale Schuurmans and Russell Greiner. Learning default concepts. In *CSSCI-94*, 1994.

G. Shackelford and D. Volper. Learning k-DNF with noise in the attributes. In *Proc. 1st Annu. Workshop on Comput. Learning Theory*, pages 97–103, San Mateo, CA, 1988. published by Morgan Kaufmann.

Geoff Towell. *Symbolic Knowledge and Neural Networks: Insertion, Refinement and Extraction*. PhD thesis, University of Wisconsin, Madison, 1991.

Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–42, 1984.

James Wogulis and Michael J. Pazzani. A methodology for evaluating theory revision systems: Results with Audrey II. In *Proceedings of IJCAI-93*, 1993.